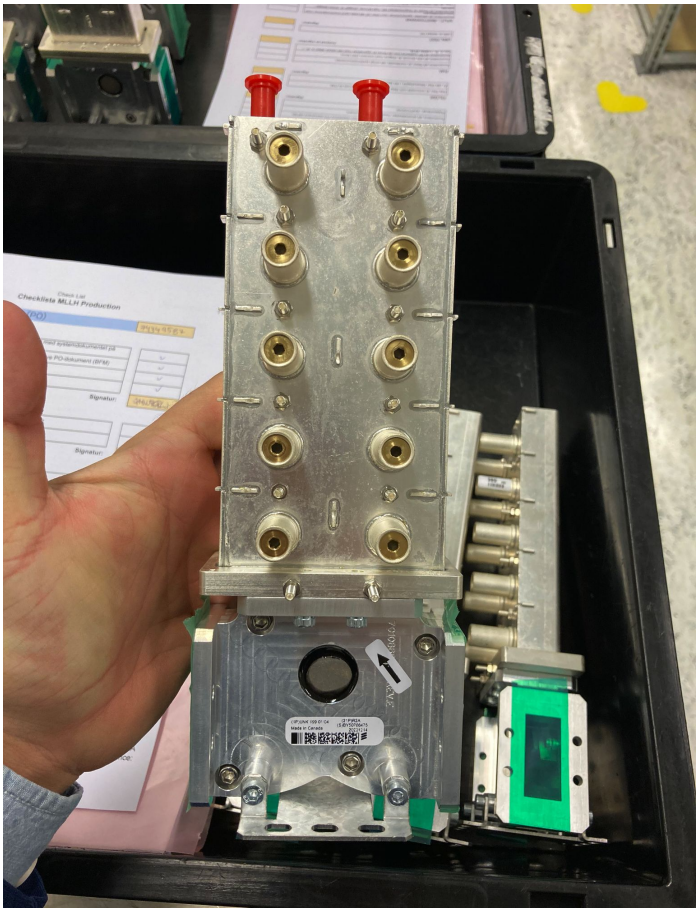
CHALMERS



Automation of Test Steps

Possible solutions for automation of manual filter tuning steps

Bachelor's thesis in Computer Science and Engineering

Dennis Ceric
Edvin Fazlagic

Automation of Test Steps
Possible solutions for automation of manual filter tuning steps

Dennis Ceric
Edvin Fazlagic

Examiner: Jonas Duregård, Chalmers University of Technology
Supervisor: Sakib Sistek, Chalmers University of Technology

Department of Computer Science and Engineering
Chalmers University of Technology / University of Gothenburg
SE-412 96 Göteborg
Sweden
Telephone: +46 (0)31-772 1000

Cover: Picture of a diplexer filter at Ericsson AB. The picture was taken by the authors.

Department of Computer Science and Engineering
Gothenburg 2021

Terminology & abbreviations

Artificial intelligence (AI): Describes the capability of machines to imitate human cognitive functions.

Artificial neural networks (ANN): A network of neurons that makes logical decisions, based on the neural network of the human brain.

Fuzzy logic: A many-valued logic system, where a variable can take on any real number between 0 and 1.

Vector Network Analyzer (VNA): A radio frequency instrument that measures performances of microwave devices.

Feed forward network: A type of ANN, where the neurons are organized in layers. Each neuron in a layer is connected to neurons in a previous layer.

Back propagation: A type of training method for ANNs, so that the network can make better and improved decisions.

Secure digital card(SD-card): A memory card that can read, store and write data.

ABSTRACT

Automation is a relevant topic in this day and age for the telecommunication industry. The purpose of this thesis was to analyze the current production flow of a tuned microwave diplexer filter at Ericsson AB, then identify possible automated solutions of the current manual filter tuning steps, as a proof of concept. Practical studies were done to analyze how all the current manual filter tuning steps were done. After all the steps were analyzed, the filter tuning steps that should be automated were clarified. Thereafter, three possible solutions were being worked on but one solution was scrapped early. The first solution being based on potentiometer controlled servo motors connected to the tuning elements of the filter, where the operator can tune a filter with potentiometers instead of manual screw drivers. This solution was further illustrated by presenting sketches of potentiometer controlled servo motors with the help of Arduino. The second solution being an AI-based robotic solution, where research was conducted on what types of solution exist today and what needs to be done in order to develop this solution. The last solution that got scrapped was based on recording the operator's screwing movement, thereafter implementing the recorded data in a robot who would execute the exact same movements. All the benefits and potential drawbacks were brought up for the proposed solutions, from a cost and quality perspective, as well as identifying which steps the proposed solutions can automate. From a short term perspective, the potentiometer based solution is the best to implement as an inexpensive and quick solution. However, as a long term solution, the AI-based solution is definitely worth investing time to work on, in order to reach full automation of all the manual steps to save time and money.

Keywords: automation, filter tuning, microwave filters, artificial intelligence, neural networks, fuzzy logic, vector network analyzer. servo motor, potentiometer, arduino

SAMMANFATTNING

Automation är ett relevant ämne inom telekommunikationsindustrin. Syftet med denna avhandling var att analysera det aktuella produktionsflödet för ett trimmat mikrovågs diplexerfilter på Ericsson AB. Därefter identifiera möjliga automationslösningar av de manuella filtertrimnings stegen som finns i nuläget, som ett koncepttest. Praktiska studier gjordes för att analysera hur alla nuvarande manuella filtertrimningssteg utfördes. Efter att alla steg analyserats, klargjordes stegen i filtertrimningen som kan automatiseras. Tre möjliga lösningar på, varav en skrotades tidigt presenterades. Den första lösningen är baserad på potentiometerstyrda servomotorer, som är anslutna till skruvarna hos filtret. Operatören kan trimma filtret med potentiometrar istället för manuella skruvmejslar. Denna lösning illustrerades ytterligare genom att presentera skisser av potentiometerstyrda servomotorer med hjälp av Arduino. Den andra lösningen är en AI-baserad robotlösning, där forskning genomfördes om vilka typer av lösningar som existerar idag, samt vad som krävs för att kunna implementera denna lösning. Den sista lösningen som skrotades handlade om inspelning av operatörens skruvrörelser, för att sedan implementera denna data i en robot som ska utföra exakt samma rörelser. Alla fördelar och potentiella nackdelar togs upp för de föreslagna lösningarna, ur ett kostnads- och kvalitetsperspektiv, samt att identifiera vilka steg de föreslagna lösningarna kan automatisera. Ur ett kortsiktigt perspektiv är potentiometerlösningen den bästa att implementera, som en snabb och billig lösning. Dock är AI-lösningen definitivt bäst på lång sikt värt att investera tid att arbeta på, för att nå full automatisering av alla manuella steg för att spara tid och pengar.

Nyckelord: automation, filtertrimning, mikrovågs filter, artificiell intelligens, neurala nätverk, fuzzy logic, vector network analyzer. servo motor, potentiometer, arduino

List of figures

PREFACE

We want to give special thanks to our supervisor Sakib Sistek from Chalmers University of Technology. Sakib supported us all through this project and we want to thank him for that and all the meetings and good discussions we had. We would also like to thank the people on Ericsson. Thank you to our supervisor Andreas Magnusson from Ericsson, who made it possible for us to get in contact with the right people and also guided us around when we were at Ericsson. Thank you to Ale Ceric for all the meetings and guidance throughout the whole project. Thank you to Klas Persson, Fredrik Jingnäs and Stefan Olsson for the meetings and feedback on our proposed solutions. Thank you to Peter Runöfors for showing us the filter tuning process. Lastly we want to thank our families for their love and support.

Table of Contents

1 Introduction

The following paragraph describes a short background about the following thesis, as well as the aim, goals, problem statement and limitations.

1.1 Background

The demand of a world more connected is ever increasing in the modern, digital age. With big telecommunications projects like 5G actively being implemented around the world, the demand of quality microwave products needs to be at a high standard.

At the Ericsson site in Borås, Sweden, the Long Haul MINI-LINK product family is one of the contributing products for increasing connectivity. These types of products have a demand of finely tuned microwave filters, in order to meet specific connectivity requirements from customers all around the world.

An increasing demand of telecommunication products, leads to an increasing demand of producing microwave filters. This will mean that the questions regarding automation will be more relevant, to identify how the production flow can be improved for Ericsson. This is what this thesis will explore, by identifying the possibilities of automation of the filter tuning process.

1.2 Aim

The purpose of this case is to present a proof of concept if it is possible to automate the filter tuning process, or in some way ease the current manual filter tuning process for an operator. Whether it be through a robotic solution, human solution or a combination of both. This thesis aims to analyze the Filter Tuning production flow and understand in which parts an automated procedure would be beneficial.

1.3 Goals

The following steps are envisioned as part of the thesis work:

- Identify all manual steps in the Filter Tuning production flow.
- Analyse and propose which manual steps would benefit from being automated, both taking cost and quality into consideration.
- Propose the best solution for automatization of the Filter Tuning process – robotic solution or combination - based on AI,VR ,AR …

1.4 Problem statement

This case is to be viewed as a proof of concept if it is possible to automate or ease some of the steps of the filter tuning process, whether it be through a robotic or human solution. The goal is to present the answer to the question “is it possible to automate or ease the current filter tuning process?”.

1.5 Limitations

- We will focus on a Proof of Concept that it is possible to automate the filter tuning process. This means that we will not implement a practical solution but rather show what kind of possible solution can be implemented.
- At Ericsson, there are both diplexer filters and “single” filters. We will focus on the diplexer filter and show that the solution is possible for the diplexer filter.
- Due to the COVID-19 pandemic, we will mostly work remotely. Some work will be done on site at Ericsson to investigate the Filter Tuning equipment and it’s practical uses, as well as what factors should be taken into consideration when the workers are tuning.

2 Theory & technical descriptions

In the following section, technical descriptions will be defined of the current manual trimware being utilized at Ericsson. As well as descriptions of potential tools and knowledge that can help automate the filter tuning steps.

2.1 Trimware at Ericsson

2.1.1 Diplexer microwave filter

A diplexer filter is a type of filter consisting of two parts: A right part and a left part, with their own specific frequency tuning targets to reach. This type of filter consists of 22 tuning elements, grouped into three different types of screws: countersunk screws, glass rods and width screws. All of the screws interact with each other, thus affecting the whole filter when the operator is tuning to find the proper frequencies for reaching a “PASS” status, which indicates that the filter is properly tuned.

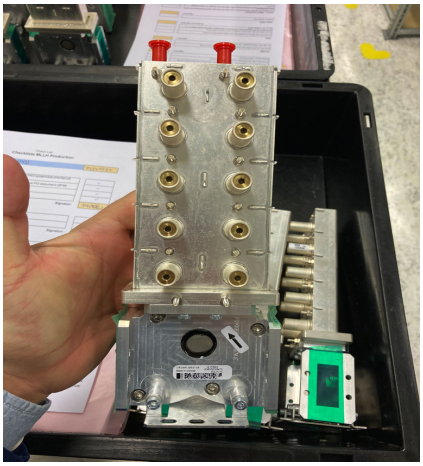


Figure 1: View over the tuning elements of a microwave diplexer filter.

2.1.2 Tuning aid tools

Multiple filter tuning tools are needed when tuning a filter, both practical tools for the operator, as well as software help.

The operator tunes filters on a work bench consisting of a fixture for mounting the diplexer filter and multiple types of automatic and manual screwdrivers for tuning the different types of screws. The fixture allows the operator to mount and tune the filter, as well as hooking calibration cables to the filter, for an internal software test program called Syborg 2017.

Syborg 2017 is an internal software test program that allows the operator to receive active feedback in real time on the operator's progress of filter tuning. With the help of this tool, the operator can tune with intuitive knowledge the filter properly, until status “PASS” is reached. The program is limited to only give feedback to the operator of the filter measures and to control if the tuned filter has reached status “PASS”.

The ideal curve to achieve in Syborg looks like the related picture. The operator reaches the requirements for filter tuning, by attempting to fit the green curve (representing the actively tuned filter) into the frequency requirements.

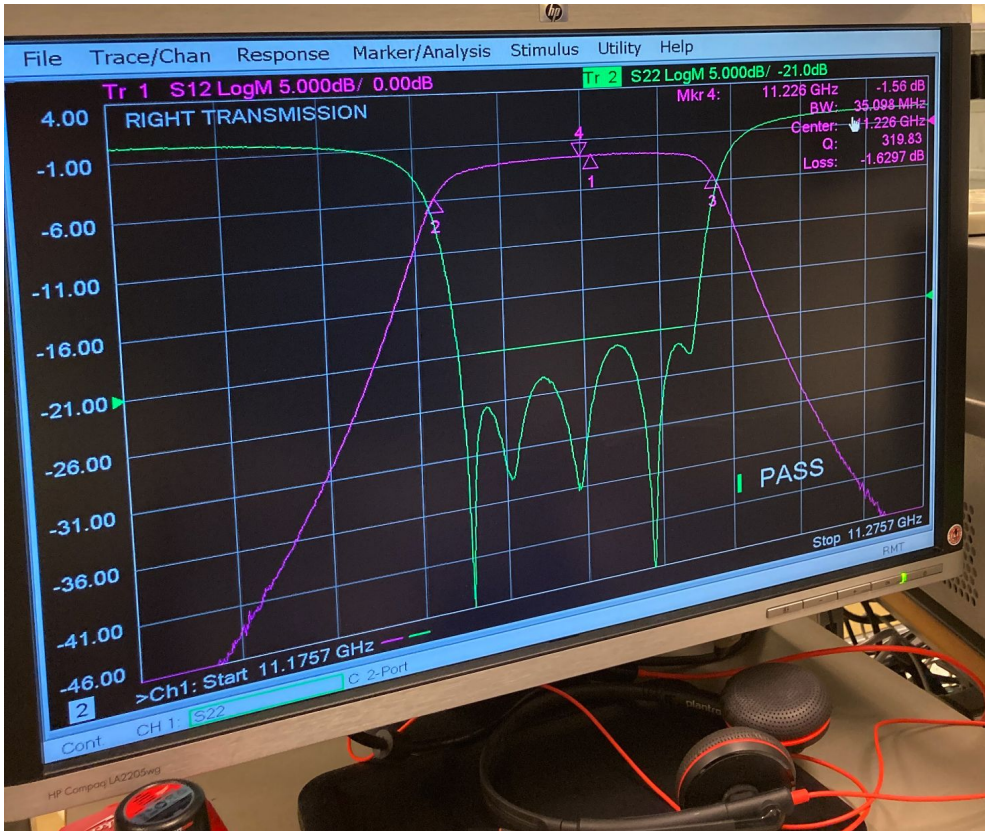


Figure 2: View of the Syborg2017 program of a tuned filter reaching status “PASS”.

During tuning of the diplexer, the operator switches between different types of screwdrivers, depending on the type of screws, as well as the frequency measures of the filter. This results with the operator switching between multiple screwdrivers consistently during tuning.

2.2 Artificial Neural Network (ANN)

An artificial neural network (ANN) is a large mathematical function inspired by the neurons in the human brain. The purpose of an ANN is to be programmed to solve problems in a way like how the human brain processes information. Several layers of neurons make up an artificial neural network. An ANN can be divided into 3 main layers of neurons (a neuron represents a variable): Input layer, output layer and a hidden layer. The input data is on the far left in the first layer, and the output or "response" is on the far right side. The hidden layers in between are used to improve the network's ability to determine the proper response, which requires empirical experiments to improve and perfect over time.

ANNs are capable of detecting patterns in non-linear data and are error-resistant. If the environment changes, they can be changed as well. ANNs are useful for a variety of

tasks, e.g clustering or function approximation [1]. ANNs require a broad training set in order to improve the algorithm's functionability.

A neural network must be trained with the same form of input data that it can understand in the later tuning in order to provide accurate answers. A popular type of ANN-architecture, is a feed forward network back propagation algorithm. It can be used for data modeling, forecasting, and pattern recognition, etc.

2.2.1 Definition of symbols & concepts for ANN

- S_0 = characteristic of a tuned filter
- Z_0 = Corresponding normalized positions for the tuning elements (screws)
- $S = f(z)$ - The tuned filter is a function of all normalized screw positions
  - Operator A: S ⇒ ΔZ, where ΔZ= normalized increment of the screws
  - Application of ΔZon tuning positions Z, leads to the tuned filter S
  - Formula: S_0 = f (Z+ΔZ)
- u =Unit describing screw adjustment increments ($u = \pm 18\ degrees$)
- $W(j)$= the values of the screw positions, specified as multiple values of u
- W_{0L}^{α} = The screw positions of the tuned filter. W_{xL}^{α} = The screw positions of the untrimmed filter, to collect learning samples. W_{0T}^{α} = The screw positions of the trimmed filter, for use in the de-trimming process and collecting "testing samples". W_{xT}^{α} = The screw positions during the time all test vectors are collected.
- For operator A, the vector couple used is (S_x, ΔZ_x), where $\Delta Z_x = Z_0 - Z_x$.
  - ΔZ_x =The difference between screw positions of a tuned filter $Z_0$and the screw positions of a detuned filter Z_x
  - S_x= Corresponding characteristic of a detuned filter
- Randomly generated vector couples (S_x, ΔZ_x) = Training vectors used to teach ANN. During this process, both learning & training vectors are used to optimize the ANN. The training vectors must be prepared in order to train the ANN
  - The input vectors used in the ANN is S_{11}(sampled characteristic of a detuned filter) and output vectors are the normalized screw deviations ΔZ_x.

2.3 Supervised Learning

Supervised learning is used in back propagation networks for ANNs. The input layer receives data, then passes on to the first hidden layer. Each layer's output will be sent forward until it reaches the output layer, which will generate the network's overall output. The discrepancy between the network output and the desired output is used to measure an error. Starting with the output layer and working backwards. [2]

2.4 Fuzzy Logic

With Fuzzy Logic, continuous values can be divided into overlapping regions. Each value can belong to different regions between 0 and 1. Eg a person could be considered both as “young” and ”middle aged”. But if the person is considered old enough to not be young at all, the person would only belong to the “middle aged” region and not “young”.

A function defines the degree to which a given value belongs to a fuzzy group.
This function can be triangular for instance, with the top of the triangle equaling 1, with x = the variable divided into Fuzzy Sets, and y = a value that indicates how much of a value belongs to a particular area. As a result, instead of just 0 or 1, the algorithm's outputs can take any value within the permitted range.

This will allow constructing a mapping of input sets to output sets, if both the input and output variables are separated into Fuzzy Sets. After that, the output values can be measured for a given set of inputs, taking into account how many of the input values are Fuzzy Sets. As a result, instead of just being bound by either 0 or 1 values, the algorithm's outputs can take any value within the permitted range [3].

2.5 Vector Network Analyzer

The purpose of a Vector Network Analyzer (VNA) is to measure the electrical behaviour of network parameters. In this case, a common parameter to measure in microwave filters are called scattering parameters (s-parameters). S-parameters are described as complex numbers, to show amplitude and/or phase of the reflection/transmission characteristics in N-by-N matrices (where N stands for number of network ports of a filter) [4]. For a two-port network, the S-parameters are described by the following matrix:

$$\begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix}$$

Figure 3: A matrix describing the S-parameters of a two-port network.

2.6 Arduino

Arduino is a company that produces microcontrollers called Arduino boards and the model used in this project is called Arduino Uno. The different models of arduino boards work in the same way but they just have different functions like bluetooth, on board wifi or more inputs/outputs. A microcontroller is a small computer processor that is mounted on the Arduino board with a variety of other components that handle inputs and outputs. So on the Arduino you can have any input that you want like a button, keyboard, potentiometer, scanner or anything that can send some information/data into the microcontroller. When the data has been sent into the Arduino the onboard processor manipulates and processes the data and decides what to do with it. Then it can send the data through the outputs to the devices you have connected to the Arduino, so you can for example control a motor, a light, a speaker or anything else. The microcontroller is open-ended both on the front and the end back. This means that you can have any inputs and control any outputs [5].

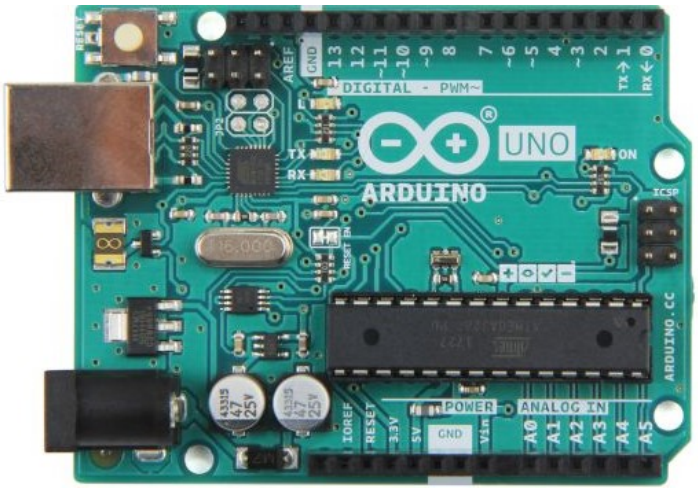


Figure 4: An Arduino UNO board.

2.7 C-programming

C-programming is one of the most popular computer programing languages and is basically used to make instructions for a computer to follow. Us humans could communicate with a computer using machine code which contains a bunch of ones and zeros but that is really complicated and therefore we use different programming languages to communicate with computers. We humans learn a programming language like C and then the programming language converts into machine code. And the machine code could be seen like individual commands that we tell the computer's processor, so all the ones and zeros translate to commands for the processor. So a programming language like C makes it easier for us humans to communicate with computers [6].

2.8 Servomotor

A servomotor is a simple self-contained electric motor that can rotate with great precision. With a servomotor you could decide the speed of how fast the servomotor rotates and how much the servomotor will rotate. The servomotor will only rotate as far as it's told to do before it stops and wait for the next command/signal. These signals that tell the servomotor how much to rotate can be sent to the servomotors with for example potentiometers and coding. The difference between a servomotor and a normal motor is that the normal motor will rotate as soon power is applied to it and stops rotating as soon as the motor doesn't get any power anymore [7].

2.9 Potentiometer

A potentiometer is a common electrical component and is a type of adjustable resistor which has three terminals. The resistance is being adjusted manually to regulate the flow of electric current, a potentiometer is basically used to regulate and vary the resistance in a circuit. There are two types of potentiometers, one who works in a linear movement “linear potentiometer” and one who works in a circular movement “rotary potentiometer”. These two types of potentiometers work in the same way, the only difference is the construction of the potentiometers and in what path the wiper of the potentiometer is able to move in [8]. To increase/decrease the resistance in a linear potentiometer the wiper is being moved in a linear path as shown in the picture below. To increase/decrease the resistance in a rotary potentiometer the wiper is being dragged in a circular path as shown in the picture below. A rotary potentiometer can be either single-turn or multi-turn, some common turning ranges are 3, 5, 10 turns but the max turning range can be even more.

Linear potentiometer

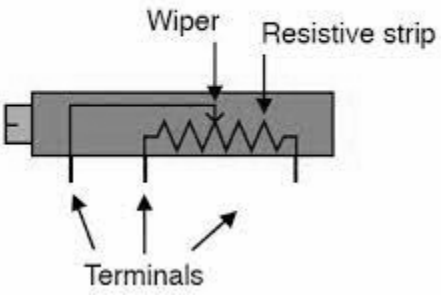


Figure 5: Illustration of a linear potentiometer.

Rotary potentiometer

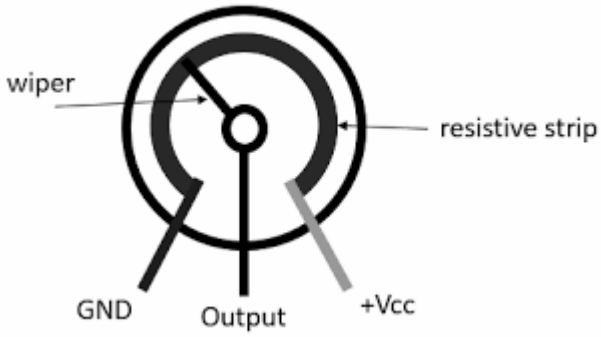


Figure 6: Illustration of a rotary potentiometer.

3 Method & implementation

In the following section, a method of evaluating the manual filter tuning steps is described. Thereafter, methods of proposed potential solutions are constructed with the help of knowledge from the theory section.

3.1 Evaluating the manual filter tuning steps

When evaluating the manual filter tuning, we oversaw the filter tuning process by an expert tuner from start to finish. The whole process was observed and recorded, so that all manual steps from start to finish can be analyzed and broken down into what main stages the operator went through during filter tuning.

3.2 Creation of the Potentiometer solution

In the following paragraph, a potentiometer based solution on controlling servo motors was suggested. This solution is not an automation based solution. This is a solution that would ease the filter tuning for an operator. The main motivation for this, is instead of manually turning the screws with screwdrivers, the tuning elements can be attached to servo motors. These servo motors are connected between the tuning elements of the filter and the potentiometers. This means that the operator can tune the filter using potentiometers instead of using several screwdrivers. This solution aids the filter tuner as a combination of a human and robotic solution.

Firstly, a virtual sketch of potentiometer controlled servo motors was done in a free software program called TinkerCAD. When our virtual coupling was tested and verified to work, a practical coupling was created.

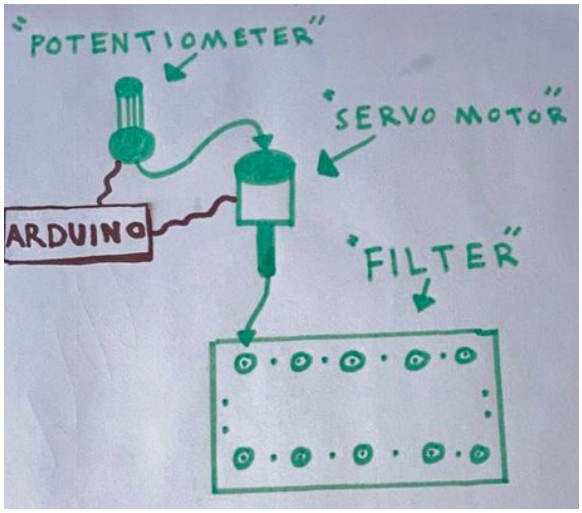


Figure 7: Illustration of how a servo motor can be connected between a filter screw, Arduino and a potentiometer.

3.2.1 TinkerCAD sketch

To make a virtual sketch of the potentiometer solution in TinkerCAD, the following components were used:

- An Arduino UNO
- Five potentiometers
- Five servo motors
- A breadboard.
- Coupling wires

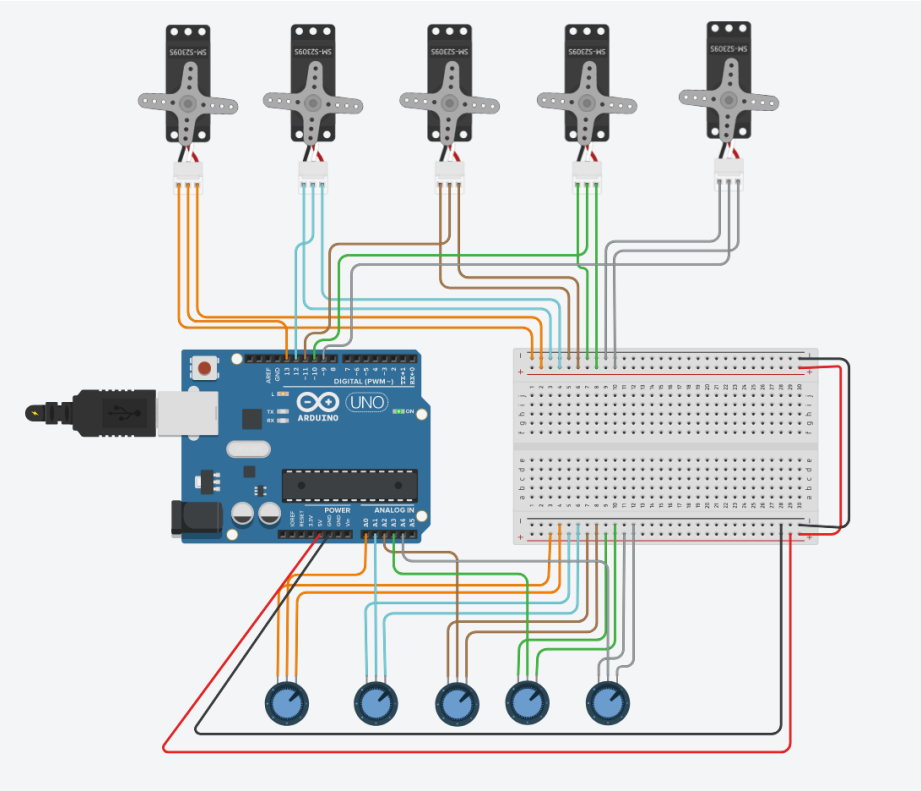


Figure 8: TinkerCAD coupling of potentiometer controlled servo motors with an Arduino

Firstly, learning was done regarding how servo motors are controlled in Arduino. This meant researching how each component worked, what type of code needs to be initiated and how to properly make the coupling work with the code.

After multiple attempts of trial and error with writing the code and testing the sketch, each servo motor was able to be controlled by its corresponding rotary potentiometer. Each servo motor in TinkerCAD is limited to a maximum turn of 180 degrees. This is due to limitations of the given components in the program.

3.2.2 Practical Arduino coupling

The practical coupling was done with only one servo motor and one rotary potentiometer. This is to show that the functionality remains the same, regardless of how many motors and potentiometers are connected. If the coupling is made possible with 1 and 5 motors and potentiometers each, then the same principle can be duplicated for the 22 tuning elements of the filter as well.

For the practical coupling, the code was prepared beforehand. The code was based on the previous code for our TinkerCAD sketch, but customly tailored for only one rotary potentiometer and motor with the following components:

- An Arduino UNO
- A potentiometer
- A servo motor
- A breadboard.
- Coupling wires

3.3 Research compilation of AI-solution

Research was conducted by analyzing a multitude of scientific reports on what possible robotic AI-automated solutions exist today, for tuning related to microwave diplexer filters. The main components researched for creating an AI-solution were based on Fuzzy logic, ANN, VNA and scientific findings from reports for filter tuning automation, by J.J Michalski. The findings were compiled to explain how each component works in order to achieve a fully automated robotic tuner.

3.4 Recording tuning element movements of an operator

A solution based on recording the exact rotation of the filter screws was suggested. The thought about this solution is that an operator tunes an untuned filter until the filter reaches status “passed”, where all the tuning movements the operator has done of the screws is recorded. The recording of how much each screw is being turned can be done by connecting each screw to a potentiometer and a servo motor, much like our first suggested solution so the screw and the potentiometer are turning mutually. When a potentiometer is connected to each screw a SD-card can be connected and the SD-card can save the data of how much each screw/potentiometer has been turned. A SD-card “Secure digital card” is a memory card that is used to read/store and write large amounts of data. After the data of how much each screw needs to be turned to reach the status “passed” has been saved, it can be uploaded/programmed into a robot who will repeat these exact movements on the same type of filter. So for example if the operator turned the first screw exactly 354 degrees, this data is being saved and implemented to a robot who will turn the first screw exactly 354 degrees on a untuned filter, and this should be implemented for each screw. This possible solution was later scrapped and the reasons for that will be brought up in the discussion segment “5.1 Selection of potential solutions”.

4 Results

In the following section, the results are presented based on the described methods in the previous section. The results describe all the manual filter tuning steps, the potentiometer based solution and the compiled research for the AI-solution.

4.1 The manual filter tuning steps

The filter tuning process is divided into three main phases. The first one being pre-tuning of the filter. then parallel tuning of the screws for the pre tuned filter and finally fine tuning.

Firstly in the pre-tuning phase, the operator prepares the chosen filter for pre-tuning, by removing red covering lids attached to it. The operator then logs in to the program Syborg 2017 and clicks on “Test UUT''. This allows the operator to scan multiple barcodes attached to the filter, so that needed filter tuning data is visibly shown. In order, the operator scans firstly a barcode on top of the filter, thereafter scans the height of the screws and left channel of the diplexer, then repeats the same process for the right channel. The operator then attaches the filter to a fixture on the working surface and connects two calibration cables with the help of a torque wrench. The right canal is connected from port C and the left canal from port A of the calibration module.

The operator then begins tuning one of the glass rods, so that a visible minimum point is shown on the Syborg graph. This height of the minimum point is then shortened until only a slight minimum point is visible, by using a manual screwdriver on one of the countersunk screws.

Secondly in the parallel tuning phase, the operator tunes and continually adjusts all of the countersunk screws, glass rods and width screws. The operator does this intuitively, by receiving live feedback from the changes of the curve when tuning the screws. This will result in multiple maximum- and minimum points formed in the curve. The goal is to eventually reach status “PASS”, where the tuned filter meets all the requirements needed.

Lastly in the fine tuning step when status “PASS'' is achieved, the operator then screws on locknuts on all of the glass rods using a screwdriver. When all of the locknuts are screwed on, the operator then checks if they are stable by lightly tapping on the glass rods. The operator then saves the data on Syborg by clicking on “store values''. The filter is then fully tuned, and placed on a wagon together with other tuned filters.

Figure 9: Flowchart of the filter tuning process divided into three phases.

4.2 Potentiometer solution

4.2.1 TinkerCAD sketch & code

The following code below was used for the TinkerCAD sketch. Only a part of the code is shown of the loop function. The full code is available in Appendix 1.

Firstly, the Servo.h library is included so that the Arduino board can control the connected servo motors. Then, variables pot1 to pot5 were declared as our potentiometers, with each variable connected to Arduino analog inputs A0 to A5. Then another set of variables valPot1 to valPot5 was declared as variables that read the values for each corresponding analog pin.

In the first function void setup(), each servomotor is connected to the digital outputs of the Arduino, thus making it possible for the motors to be controllable. Lastly in the void loop() function, each potentiometer is able to turn between 0 and 180 degrees.

```
void loop()
{
 //read the values of potentiometers
 valPot1 = analogRead(pot1);
 valPot1 = map (valPot1, 0, 1023, 0, 180); //value between 0 and 180
 servo1.write(valPot1);
 delay(15);

 valPot2=analogRead(pot2);
 valPot2=map(valPot2, 0, 1023, 0, 180);   //value between 0 and 180
 servo2.write(valPot2);
 delay(15);

 valPot3 = analogRead(pot3);
 valPot3 = map (valPot3, 0, 1023, 0, 180); //value between 0 and 180
 servo3.write(valPot3);
 delay(15);

 valPot4 = analogRead(pot4);
 valPot4 = map (valPot4, 0, 1023, 0, 180); //value between 0 and 180
 servo4.write(valPot4);
 delay(15);

 valPot5 = analogRead(pot5);
 valPot5 = map (valPot5, 0, 1023, 0, 180); //value between 0 and 180
 servo5.write(valPot5);
 delay(15);
}
```

4.2.2 Practical Arduino coupling & code

After testing and verifying that the TinkerCAD sketch was working, the next step was to create a practical coupling. The coupling was done in one of the laboratory rooms at Chalmers University.

For the practical coupling, the code was prepared beforehand. Once again, only a part of the code is shown of the variable declaration and setup function. The full code is available in Appendix 2. The code was based on the previous code for our TinkerCAD sketch, but customly tailored for only one potentiometer and motor with the following components:

- An Arduino UNO
- A potentiometer
- A servo motor
- A breadboard.
- Coupling wires

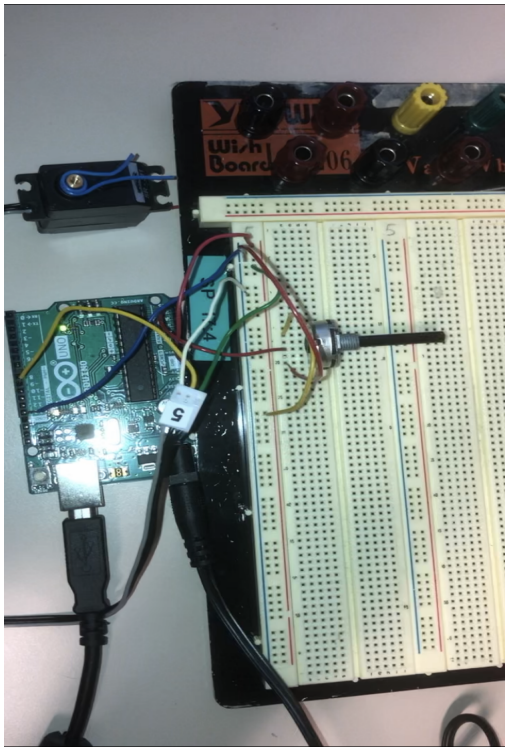


Figure 10: Practical Arduino coupling of a potentiometer controlled servo motor.

```
#include <Servo.h>

Servo servo1;//define our servomotors

int pot1 = A0;//define our potentiometers

int valPot1;// variables to read the values from the analog pin


void setup()
```

```
{
  //attach our servos to pins
  servo1.attach(13);

}
```

4.3 Research compilation of AI-solution

4.3.1 Fuzzy Logic

In their analysis, Miraftab and Mansour [3] demonstrated how a Fuzzy Logic Controller (FLC), which is made up of several, smaller Fuzzy Logic Systems (FLS), could replace a human operator.

When a robot is used to tune a filter automatically, it treats each FLS as a Fuzzy Set. It will choose the Fuzzy Set that is most similar to the input scenario. This collection corresponds to a FLS, which uses the input to select an appropriate trimming instruction. Applying this instruction provides a new scenario. This is then repeated until the filter meets the trimming requirement, or until a certain number of iterations have been completed without the fine-tuning requirement being met. In the latter case, the fine-tuning would be done manually by an expert.

A human expert is enlisted to fine-tune a filter when developing a FLS. The FLS receives the filter's S_{11}frequency response and compares it to the desired S_{11}frequency response. After that, the human expert can generate an output in the form of an adjustment to a specific parameter. In a FLS, the human output to a particular input is registered and saved as an input/output pair. This method generates a number of FLS by providing the human expert with a variety of starting scenarios.

During tuning, each FLS is handled as a fuzzy set. It will choose the fuzzy set that most closely resembles the input scenario. This collection corresponds to a FLS that uses the input to select an appropriate tuning instruction. This instruction would result in a new scenario. This is repeated until the filter meets its specifications, or until a certain number of iterations have been completed without the filter meeting its specifications. In the latter scenario, the filter is manually fine tuned by a human expert and converts the tuning into a new FLS.

4.3.2 Usage of a VNA together with ANN

Michalski [9] described a method for using a VNA (Vector Network Analyzer) when fine-tuning a cavity filter using an ANN. The network was trained along with a VNA that showed how far a screw had deviated from its correct location, based on the S_{11}parameter as an input (the S_{11}parameter measures the reflection characteristics of the sent signal). The network then returned as an output how much each screw must be turned, in order to reach proper tuning.

When the ANN reads inputs of the S-characteristics, the screw deviations ΔZ is generated in its output. The filter is then tuned after the corrections on the screw positions. The screw deviations are visualized in a 3D-space, where the distance to origo (the zero value) represents a properly tuned filter, without screw deviations. All points in this 3D-space define the "search space", with the goal of the ANN to reach origo.

To avoid manual fine tuning, Michalski [10] used the same neural network and trained it on multiple filters. According to Michalski's findings: When the ANN was trained on multiple filters, the network tuned so well that no manual fine tuning was needed.

4.3.3 Method for creating new learning vectors by Michalski

The following method described below is the step by step methodology used by Michalski for creating learning vectors [9]:

1. Use a properly tuned filter, manually tuned by an expert
2. Read the tuned characteristic sample values: $S_0(i);\ i = 1,2 \ldots M.\ (M = 512\ points\ of\ complex\ S - characteristic$ and corresponding screw positions $W_0(j), j = 1,2 \ldots N$ of the tuned filter
3. Randomly "undo" the trimmed filter manually according to formula (1) & (2):

$$W(j) = W_0(j) + \Delta W(j) \qquad (1)$$

$$\Delta W(j) = RND[2 \cdot K] - K \qquad (2)$$

K =Maximum screw deviation, a multiple value of u
$RND[X]$ =an operator returning a randomized integer value between 0 and X.
$W_0(j)$ =values of the screw positions of the tuned filter.
$\Delta W\ \ (j)$ =values of the screw positions of the detuned filter.
N = The amount of tuning elements of the filter.

4. Read the corresponding S_x as a sample of the input vector and the normalized screw increment ΔZ_x. This characteristic defines the input vector in the ANN learning process.
5. Store S_x as a sample of the input vector and ΔZ_x as a sample of the output vector. Formula (3) is a condition that performs the transformation of the screw increment to the domain from 0 to 1

$$\Delta Z_x(j) = \frac{\Delta W_x(j)}{2K} + 0.5 \qquad (3)$$

6. Repeat step 3 to create a new learning vector

The procedure helped in the development of obtaining new learning vectors and the acquisition of vector pairs for the ANN, in order to develop it further. Vector pairs were

obtained by defining the relation S ⇒ ΔZ_x, with the position of the trimmed screws as the reference position for preparing the training vectors, as well as during the tuning process. By using the vector pair in the ANN-learning process, the ANN system was created by generating relative screw deviations ΔZ_x,for untrimmed filter characteristic S (with respect to the positions of the screws on the trimmed filter). This means if the ANN responds with "zero vector"ΔZ_x, that means the tuning goal has been met.

Michalskis experiments were done to a 17-cavity diplexer filter. As a result of applying detuned reflection characteristic$S_{11}(i)$ to ANN input, the network responded with its output $\Delta Z \quad (j)$. Using this, the screw deviation $\Delta W \quad (j)(u)$ was determined. The goal was to obtain a "zero response" on ANN output. The filter characteristic S_0 was obtained after the last screw has been correctly tuned, and the filter is properly tuned with screw deviation [u] = 0 for each screw.

4.3.4 Shortening filter tuning time with Michalski's method

With their proposed method and the help of a filter tuning robot, they succeeded in reducing the filter tuning time from 20-30 minutes to only 2 minutes. The performance is depended on if an optimal topology of the ANN has been created, which has to be decided empirically with trial and error, in order to optimize the topology [11]

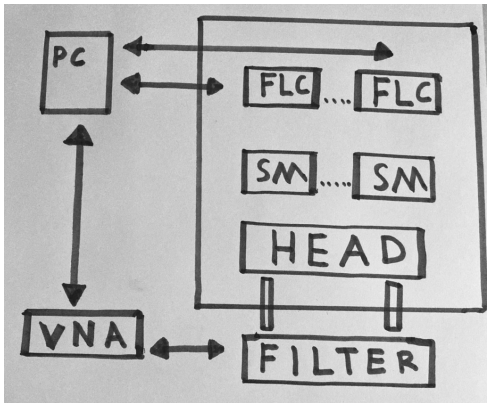


Figure 11: Illustration of the components used for robotic filter tuning automation.

5 Discussion

In this section, discussions about the results and method are brought up regarding the pros and cons of solutions, the ethics and sustainability and the decided selection of the potential solutions and what types of steps they potentially can automate.

5.1 Selection of potential solutions

In this section, we came to a conclusion of which solutions we decided to keep working on, as well as a solution that became scrapped. Numerous different solutions were presented to Ericsson, in order to help them find the best fitting solution they would be interested in implementing. We had a presentation, where three possible solutions were presented. We gave our input on which solutions we thought were the best ones and they gave us their inputs.

We came to an agreement that we should keep working on the AI-based solution and the potentiometer-controlled solution. The solution based on recording how much an operator turned each screw and then implementing that data to a robot who would repeat the exact same moment was scrapped. This solution was scrapped because we didn't see that the solution would achieve a perfectly tuned filter every time because we don't know if every type of filter screws need to be turned the exact same or if they could vary. Another problem was that Ericsson has up to hundreds of different types of filters which means that the recording process would have to be repeated for every type of filter.

5.2 Pros & Cons of the proposed solutions

For the potentiometer solution, we see that it’s possible to automate the first 2 tuning steps mainly: The pre-tuning and the parallel tuning. The benefits will be the following from a cost and quality perspective:

- Tuning the potentiometers will simplify the tuning process for the operator, in comparison to the current tuning method.
- The learning process for tuning filters for beginners is simplified.
- Only finger movements are required for filer tuning.
- Decreases the risk for repetitive strain injuries for the operator’s wrists, as the operator won't have to switch between multiple screwdrivers.
- The operator can tune multiple screws at the same time, as the operator is not limited to only tune one screw at a time anymore. With a skilled operator that has intuitive filter tuning knowledge, this can lead to a reduction of time needed when tuning a filter.
- An inexpensive investment for implementing this solution.

A potential obstacle however is for the last fine-tuning step, when applying the locknuts to the glass rods. For this step, either self locking screws need to be implemented to the

glass rods, or be screwed manually by the operator, as in the current manual filter tuning steps.

From a cost perspective, there are some aspects that can not be put into worth. An example of this is decreasing the risk for strain injuries. It may be possible to do a monetary cost analysis on aspects such as time saved, the amount of filters being tuned, etc, but not when taking the health of an operator into account.

For the AI-based solution, we believe it's possible to automate all filter tuning steps. Taking consideration to our presented research, the filter tuning process could be automated from beginning to the end with enough training. The presented benefits and cons are the following:

- Eliminates the manual filter tuning learning process for an operator
- Much more time-effective than manual filter tuning
- Increases work productivity per filter, thus increasing the production of the amount of filters
- Decreases production costs per unit
- Dependency on highly skilled filter tuning operators is decreased.

The main downside of the AI-based solution is that the solution will be time consuming to implement practically. Practical implementation will require frequent testing, experiments and feedback in order to develop and adapt a robotic filter tuner for automation. Especially when adjusting the topology of an ANN through trial and error [6]. Despite this, we believe the benefits for automation of all filter tuning steps far outweigh the main downside.

5.3 Ethics

Artificial intelligence is a frequently discussed topic regarding ethics. The most relevant ethical standpoint in this case, is how a potential implementation of a filter tuning robot will affect the currently employed filter tuning operators. A potential fear from the operators might be that they become "obsolete" when implementing an AI-based filter tuning robot. This is an important topic for Ericsson to discuss internally regarding the potential consequences, if they decide to move ahead with this solution.

5.4 Sustainability

According to the UN Agenda 2030 goals [12], sustainability is defined on three key principles: social-, environmental- and economic sustainability. From an economical and social perspective, increased production productivity and time spared can contribute to economic growth for Ericsson, as well as increased well being for the health of the workers when decreasing the risk for work related injuries. Environmentally, implementation of either solution will not lead to a significant negative drawback. The only environmental impact might be new material that has to be constructed, for creating either practical solution.

6 Conclusion

The purpose of this case was to analyze all of the manual filter tuning steps in the current production flow. The results have shown step-by-step how the operator tuned and untuned diplexer filter from start to finish to meet the given tuning goals. To further illustrate the process, the filter tuning methodology has also been categorized into three main phases the operator goes through during the tuning process.

The presented solutions have been compared to the manual filter tuning steps, to identify what types of steps are possible to automate. From our conducted research and experiments, it is possible that both the AI-based solution and potentiometer controlled solution can help automate all the steps of the current tuning method. The benefits and potential drawbacks of possible implementation for each solution have been presented

6.1 Future recommendations

Lastly for proposing the best fitting solution in Ericssons case. As an inexpensive short term solution, we recommend the potentiometer solution. It is easier and faster to implement practically. But in the meanwhile, we recommend that Ericsson work on the AI-solution over a long term time period, as stated before, the process of implementing this solution is time-consuming and requires work. When the AI-solution is implemented practically, it will definitely pay off over the long term as a great investment!

Bibliography

[1] I.A Basheer and M Hajmeer. “Artificial neural networks: fundamentals, computing,design, and application”. In:Journal of Microbiological Methods43.1 (2000). Neural Com-putting in Microbiology, pp. 3–31.

[2] P. Norvig, S.Russell. Artificial Intelligence A Modern Approach. 3rd ed. 2014.

[3] V. Miraftab and R.R. Mansour. “Tuning of microwave filters by extracting human experience using fuzzy logic”. In: Microwave Symposium Digest, 2005 IEEE MTT-S International. June 2005.

[4] Google, "S-parameters," 24 April 2021. [Online]. Available: https://www.microwaves101.com/encyclopedias/s-parameters

[5] Arduino2021, Software. https://www.arduino.cc/en/Main/Software

[6] J. Skansholm. C från början. 2nd ed. 2017.

[7] R. Firoozian. Servo Motors and Industrial Control Theory. 2nd ed. 2014.

[8] B. Graham. Introductions to Biomechatronics. 2014. pp. 27-28.

[9] J. J. Michalski. “Artificial neural networks approach in microwave filter tuning,” Progress In Electromagnetics Research M, Vol. 13, 173–188, 2010.

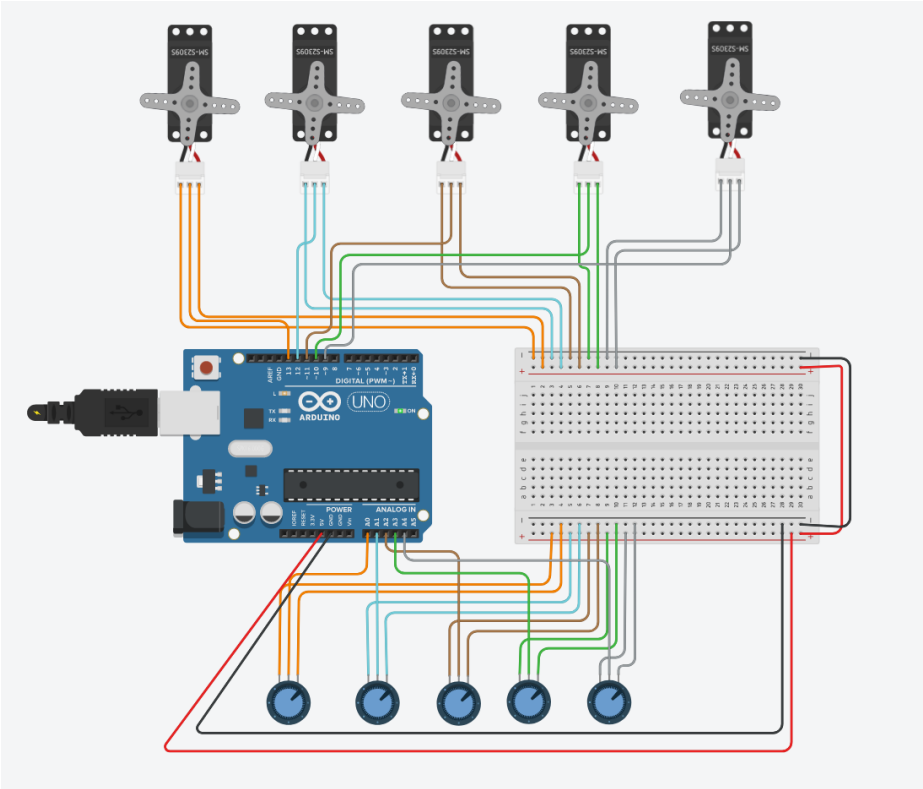
[10] J.J. Michalski. “Artificial neural network algorithm for automated filter tuning with improved efficiency by usage of many golden filters”. In: Microwave Radar and Wireless Communications (MIKON), 2010 18th International Conference on. June 2010, pp. 1–3.

[11] J.J. Michalski, T. Kacmajor, J. Gulgowski and M. Mazur. “Consideration on artificial neural network architecture in application for microwave filter tuning”. In: vol. 7.3. 2011, pp. 271–275.

[12] United Nations, “THE 17 GOALS”, May 2021. [Online]. Available: https://sdgs.un.org/goals

Appendix

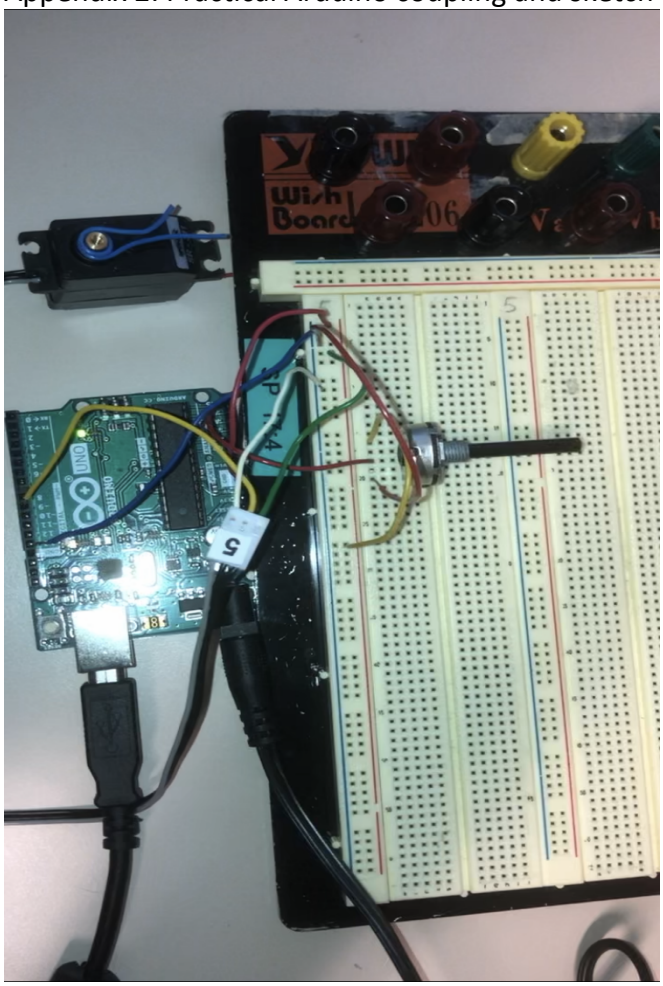
Appendix 1: TinkerCAD sketch and code of coupling with five potentiometers



```
#include <Servo.h>

Servo servo1;//define our servomotors
Servo servo2;
Servo servo3;
Servo servo4;
Servo servo5;

int pot1 = A0;//define our potentiometers
int pot2 = A1;
int pot3 = A2;
int pot4 = A3;
int pot5 = A4;

int valPot1;// variables to read the values from the analog pin
int valPot2;
int valPot3;
int valPot4;
int valPot5;

void setup()
```

```
{
  //attach our servos to pins
  servo1.attach(13);
  servo2.attach(12);
  servo3.attach(11);
  servo4.attach(10);
  servo5.attach(9);
}


void loop()
{
  //read the values of potentiometers
  valPot1 = analogRead(pot1);
  valPot1 = map (valPot1, 0, 1023, 0, 180); //value between 0 and 180
  servo1.write(valPot1);
  delay(15);

  valPot2=analogRead(pot2);
  valPot2=map(valPot2, 0, 1023, 0, 180);   //value between 0 and 180
  servo2.write(valPot2);
  delay(15);

  valPot3 = analogRead(pot3);
  valPot3 = map (valPot3, 0, 1023, 0, 180); //value between 0 and 180
  servo3.write(valPot3);
  delay(15);

  valPot4 = analogRead(pot4);
  valPot4 = map (valPot4, 0, 1023, 0, 180); //value between 0 and 180
  servo4.write(valPot4);
  delay(15);

  valPot5 = analogRead(pot5);
  valPot5 = map (valPot5, 0, 1023, 0, 180); //value between 0 and 180
  servo5.write(valPot5);
  delay(15);
}
```

Appendix 2: Practical Arduino coupling and sketch



```
#include <Servo.h>

Servo servo1;//define our servomotors

int pot1 = A0;//define our potentiometers

int valPot1;// variables to read the values from the analog pin

void setup()
{
  //attach our servos to pins
  servo1.attach(13);

}


void loop()
{
  //read the values of potentiometers
```

```
  valPot1 = analogRead(pot1);
  valPot1 = map (valPot1, 0, 1023, 0, 180); //value between 0 and 180
  servo1.write(valPot1);
  delay(15);

}
```

Appendix 3: Video of Practical Arduino coupling

https://youtube.com/shorts/mrjMCvzYmVQ?feature=share