UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
E scola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

Work-flux optimization for the use of GRASP algorithm in quasi-real-time

Master Thesis
submitted to the Faculty of the
Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya
by

Mireia López Barrón

In partial fulfillment
of the requirements for the master in
(Write the name of your Master) **ENGINEERING**

Advisor: name of the advisor
Barcelona, Date XXXXX

Contents

List of Figures 4

List of Tables 4

1 Introduction 7
1.1 Gantt Diagram . . . 7
1.2 Topic . . . 7

2 State of the art of the technology used or applied in this thesis: 8
2.1 Topic . . . 8
2.2 Topic . . . 8

3 Methodology 9

4 List of requirements 10

5 Research, plan and adapt 12
5.1 Web services development . . . 12
5.1.1 ACTRIS–EARLINET service . . . 12
5.1.2 AERONET service . . . 13
5.2 Matlab controller development . . . 15
5.2.1 Create script configuration files and output files . . . 15
5.2.2 Automatize MATLAB repository for needed data files . . . 15
5.2.3 Adapt old scripts to new file formats . . . 15
5.2.4 Execute MATLAB scripts from C# . . . 15

6 Architecture 17
6.1 MVVM . . . 17
6.1.1 Observable Object . . . 18
6.1.2 Relay Command . . . 18
6.2 SOLID principles . . . 19
6.2.1 Single Responsibility Principle . . . 19
6.2.2 Open/Closed Principle . . . 19
6.2.3 Liskov Substitution Principle . . . 19
6.2.4 Interface Segregation Principle . . . 20
6.2.5 Dependency Inversion Principle . . . 20
6.3 Project architecture . . . 20

7 Graphic User Interface (GUI) design 22

8 Implementation 23
8.1 Set up development enviroment . . . 23

9 Testing 25

10 Results 26

11 Conclusions 28

References 29

Appendices 30

List of Figures

1 Project's Gantt diagram 7
2 mvvm 17
3 mvvm 20
4 Prototype setup 26

Listings

List of Tables

1 AERONET Data Products and Extensions 13
3 This is the caption 26

Revision history and approval record

Revision	Date	Purpose
0	10/12/2025	Document creation
1	dd/mm/yyyy	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
[Student name]	
[Project Supervisor 1]	
[Project Supervisor 2]	

Written by:		Reviewed and approved by:	
Date	dd/mm/yyyy	Date	dd/mm/yyyy
Name	Xxxxxxx yyyyyyy	Name	Zzzzzzz Wwwwwww
Position	Project Author	Position	Project Supervisor

Abstract

Every copy of the thesis must have an abstract. An abstract must provide a concise summary of the thesis. In style, the abstract should be a miniature version of the thesis: short introduction, a summary of the results, conclusions or main arguments presented in the thesis. The abstract may not exceed 150 words for a Degree's thesis.

1 Introduction

An Introduction that clearly states the rationale of the thesis that includes:

1. Statement of purpose (objectives).
2. Requirements and specifications.
3. Methods and procedures, citing if this work is a continuation of another project or it uses applications, algorithms, software or hardware previously developed by other authors.
4. Work plan with tasks, milestones and a Gantt diagram.
5. Description of the deviations from the initial plan and incidences that may have occurred.

The minimum chapters that this thesis document should have are described below, nevertheless they can have different names and more chapters can be added.

1.1 Gantt Diagram

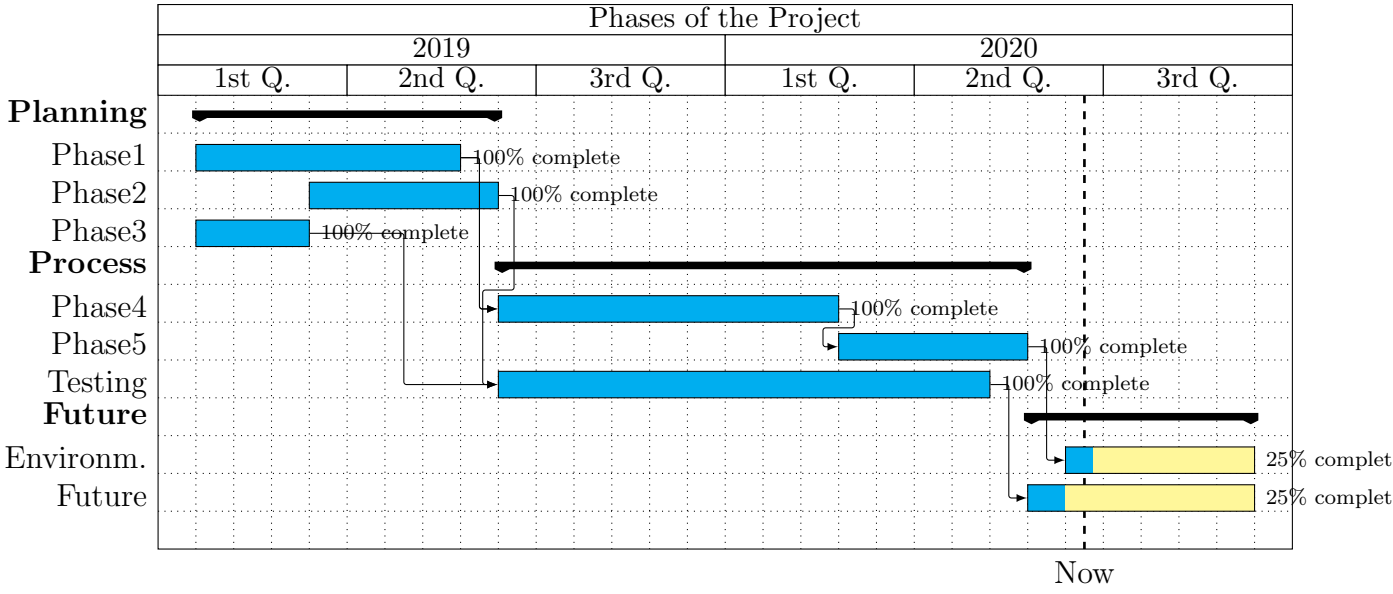

Figure 1: Gantt diagram of the project

For more information read the manual [1] of Skala.

1.2 Topic

2 State of the art of the technology used or applied in this thesis:

A background, comprehensive review of the literature is required. This is known as the Review of Literature and should include relevant, recent research that has been done on the subject matter.

2.1 Topic

Here you have a couple of references about LaTeX [3] and electrodynamics [2].

2.2 Topic

3 Methodology

In order to create a new application , it is necessary to follow the following steps, to create a strong foundation.

- Define a list of requirements
- Research, plan and adapt
- Design the architecture
- Design the graphical user interface (GUI)
- Set up development environment
- Implement
- Test
- Deployment
- Maintenance

For the development of the application, it has been decided to use an hybrid structure, using Object Oriented Programming (OOP) with C# and MATLAB scripting.

The main advantages of using C# with OOP are that it allows the code to be reusable, organized and easy to maintain [7]. In this project, it has been used for the development of the Graphical User Interface and the controllers for both, web services and MATLAB script management.

On the other hand, MATLAB provides a specialized environment for numerical calculations, simulations, and complex data analysis[8]. Working directly with matrixes and arrays makes the data processing and visualitzation easier and more efficient.

4 List of requirements

This first step is crucial for the correct result. It is needed to define a clear and well defined list of requirements. This defines the main scope of the project, the different futures to be implemented and prevents from building something that is not desired. This project, need to be able to fulfill the following requirements:

1. Application can be executed in CALCULA operating system (Linux)
2. Download measurements from the internet
   (a) Download measurements from a defined date range
   (b) Download measurements from a defined location
   (c) Download measurements from the Actris-Earlinet Data Portal [5]
      - Download ELPP products
      - Download Optical products
   (d) Download measurements from the Aeronet web [6]
      - Download Raw Almucantar Sky Scan Radiance measurements
      - Download Raw Hybrid Sky Scan Radiance measurements
      - Download Raw Principal Plane Sky Scan Radiance measurements
      - Download Raw Polarized Principal Plane Sky Scan Radiance and Degree of Polarization measurements
      - Download Raw Polarized Almucantar Sky Scan Radiance and Degree of Polarization measurements
      - Download Raw Polarized Hybrid Sky Scan Radiance and Degree of Polarization measurements
   (e)
3. Filter Earlinet measurement files depending on the the data type
   (a) 002, 008
   (b) 007
4. Execute pre-execution:
   (a) Read data of all downloaded files that are used as inputs for grasp algorithm
   (b) Preview data in order to choose a correct value for minimum and maximum heights
   (c) Obtain available configurations for the selected measure ID depending on available data
   (d) Choose a configuration

   (e) Generate a .sdat file that stores all data for selected measure ID and heights

   (f) Create .yml configuration file containing corresponding .sdat file name and choosen configuration

5. Execute the GRASP algorithm with corresponding configuration .yml file and .sdat file

6. Give the user the possibility to plot the different results after executing the algorithm

   - Detect all available measure IDs folders in output directory
   - Generate figure data files .mat from GRASP output data in selected measure ID output directory
   - Plot data from .mat files

7. Application can work with project/workspaces

   (a) User can create, open and import a project

   (b) Downloaded data from web services are stored in the project folder

   (c) Results of the GRASP algorithm are stored in the project folder

8. Application can have different configurations

   - Repository directories can be modified for both, earlinet and aeronet downloaded files
   - Ouput directory can be modified
   - GRASP installation directory can be modified: this path can be different for each user

5 Research, plan and adapt

This step helps optimize both time and resources by preventing unnecessary work. Since this project builds upon a previous one, it is not required to develop new code for certain application requirements, but to adapt it to the new needs. Furthermore, the availability of APIs for downloading measurements from the ACTRIS–EARLINET Data Portal [5] and AERONET web application [6] provides a convenient and efficient solution that can significantly reduce development effort.

5.1 Web services development

5.1.1 ACTRIS–EARLINET service

The ACTRIS-EARLINET Data Portal provides a REST API for downloading measurements [9]. This web, shows all the different requests that are supported by the API, gives examples of how to use them and the expected response.

Having all this information available is really helpful, since it allows to localize the requests that are needed for the project requirements.

After testing and comparing the different options and the obtained responses, it becomes clear that the best option is to use the "/products/downloads" endpoint, which allows to download the measurements in a compressed format, filtering the data by type, date and station name.

Even if other configuration parameters are available (measurementID, wavelength and opticaltype), it is not necessary to use them for the project requirements.

Once the behaviour is understood, a class is created to handle the requests and responses of the API.

```
1  public class EarlinetService{
2      string baseUrl = "https://api.actris-ares.eu/api/services/restapi/";
3      public virtual System.Threading.Tasks.Task<bool>
           DownloadProductByDateRangeAsync(string type,string FromDate,
           string ToDate,string outputFolder){
4          return DownloadProductByDateRangeAsync(type,FromDate,ToDate,
               outputFolder,System.Threading.CancellationToken.None);
5      }
6
7      public async Task<bool> DownloadProductByDateRangeAsync(string type,
           string FromDate,string ToDate,string outputFilePath,System.
           Threading.CancellationToken cancellationToken = default){
8          try{
9              string url = $"{baseUrl}products/downloads?kind={type}&
                   fromDate={FromDate}&toDate={ToDate}&stations=brc";
10             using (HttpClient client = new HttpClient()){
11                 HttpResponseMessage response = await client.GetAsync(url
                       );
12                 response.EnsureSuccessStatusCode();
13                 using (var fs = new FileStream(outputFilePath,FileMode.
                       Create,FileAccess.Write,FileShare.None)){
```

```
                    await response.Content.CopyToAsync(fs);
                }
                System.IO.FileInfo f = new FileInfo(outputFilePath);
                if (f.Length <= 1048)
                    return false;
                return true;
            }
        }
        catch (Exception ex){
            return false;
        }
    }
}
```

For the creation of this class it has been taken into account the fact that the DownlaodProductByDateRangeAsync method can be used in different points of the program with different configurations.It has also been implemented in such a way that adding new functionalities is easy using the same design pattern.

5.1.2 AERONET service

The AERONET web does not provide a REST API portlet, but it includes web servicie documentation that helps the implementation of a costume service.

The amount of files to be downloaded in this website is larger thant the one in Earlinet, and its documentation can be found in three different sections of the web: For the

Table 1: AERONET Data Products and Extensions

Category	Product Description	Extension
Optical Depth [10]	Aerosol Optical Depth (AOD): (descripcion de los datos)	.lev15
	Spectral Deconvolution Algorithm (SDA): (descripcion de los datos)	.ONEILL_lev15
Aerosol Inversions [12]	Inversion products: (descripcion de los datos)	.all
Raw Products Optical Depth [11]	Raw Almucantar: (descripcion de los datos)	.alm
	Raw Polarized Almucantar: (descripcion de los datos)	.alp

creation of this class it has been taken into account the fact that the DownlaodProductByDateRangeAsync method can be used in different points of the program with different configurations.It has also been implemented in such a way that adding new functionalities is easy using the same design pattern. For clarity and organization, it has been created

the DataType Enum, since the different data types need different Url content in order to be downloaded.

```
public class AeronetService{
    private string baseUrl = "https://aeronet.gsfc.nasa.gov/cgi-bin/";
    public async Task DownloadDataAsync(string destinationFile,string
        url){
        try{
            using (HttpClient client = new HttpClient()){
                var response = await client.GetAsync(url);
                response.EnsureSuccessStatusCode();

                var content = await response.Content.
                    ReadAsByteArrayAsync();
                await File.WriteAllBytesAsync(destinationFile,content);
            }
        }
        catch (Exception ex){
            Console.WriteLine($"Error downloading data: {ex.Message}");
        }
    }

    public string BuildUrl(DataType _dataType,DateTime startDate,
        DateTime endDate,string productType = "",string site = "",string
        product = "",string AVG = "",bool isEnabled = false){
        switch (_dataType){
            case DataType.AerosolInversions:
                if (isEnabled)
                    return BuildUrlAerosolInversions(startDate,endDate,
                        productType,site,product,AVG);
                else
                    return BuildUrlAerosolInversions(startDate,endDate);

            case DataType.OpticalDepth:
                if (isEnabled)
                    return BuildUrlOpticalDepth(startDate,endDate,
                        productType,site,AVG);
                else
                    return BuildUrlOpticalDepth(startDate,endDate,
                        productType);

            case DataType.RawProductsOpticalDepth:
                if (isEnabled)
                    return BuildUrlRawProductsOpticalDepth(startDate,
                        endDate,productType,site,AVG);
                else
                    return BuildUrlRawProductsOpticalDepth(startDate,
                        endDate,productType);
        }
        return "";
    }

    {...} //BuildUrl methods can be found in GitHub repository
}
```

In this application, this class is only used to download the data files listed above, but other types could be downloaded by calling only the DownloadDataAsync and BuildUrl methods in a line.

5.2 Matlab controller development

As it was mentioned in the introduction, this project is a second attempt to offer a solution of the same problem.

While the first application proposed a complete implementation in MATLAB, the current approach differs. However, since many requirements are shared between both, the previously developed code has been adapted to the new hybrid system in order to be executed with C#. In order to do that, different actions have been taken.

5.2.1 Create script configuration files and output files

Since the project is hybrid, it is necessary to create a configuration file for the script that will be executed in C#. This file will contain the parameters needed to execute the script. In the previous approach, this files were not needed, since GUI introduced parameters were being stores in the MATLAB workspace. The use of these files, allow the communication between the two different languages. Configuration files allow the different scripts to obtain the information and parameters that the user introduced, while the output files allow the application to know the corresponding results and notify the user about it.

5.2.2 Automatize MATLAB repository for needed data files

In the older version, it was needed to manually download the data files from the different websites and at the respective files in two concretes repositories. Files downloaded from Aeronet site were located in a folder inside the Matlab Project called /repository/AERONET/. Files downloaded from Earlinet site were located in a folder inside the Matlab Project called /repository/LIDAR/ and inside this folder, a folder for each measurementID was created containing all the data files corresponding to that measure. In order to automatize this process, a script has been created that downloads the data files from the AERONET website and stores them in a local repository.

In order to be able to automatize the process of downloading and giving the application the capacity of work with different workspaces, the repository location has been modified and automatized inside the application. The corresponding file is saved as configuration parameter in both, application and script.

5.2.3 Adapt old scripts to new file formats

5.2.4 Execute MATLAB scripts from C#

In order to execute the MATLAB scripts from C#, it is needed to create a method that uses the ProcessStartInfo class, that can be imported as a library, to execute the MATLAB

script. The following example shows a method called RunMatlabScript that takes a string parameter containing the path to the script to be executed. The method uses the Process class to execute the script and it redirects the standard output and standard error to the console.

```
public static void RunMatlabScript (string scriptPath){
    try{
        ProcessStartInfo startInfo = new ProcessStartInfo{
            FileName = "matlab",
            Arguments = $"-batch␣\"run('{scriptPath}')\"",
            RedirectStandardOutput = true,
            RedirectStandardError = true,
            UseShellExecute = false,
            CreateNoWindow = true
        };
        using var process = new Process { StartInfo = startInfo };
        process.OutputDataReceived += (s,e) =>{
            if (e.Data != null)
                Messenger.Default.Send<string>("WriteMatlabOutput",e.
                    Data);
        };
        process.ErrorDataReceived += (s,e) =>{
            if (e.Data != null)
                Messenger.Default.Send<string>("WriteMatlabErrors",e.
                    Data);
        };
        process.Start();
        process.BeginOutputReadLine();
        process.BeginErrorReadLine();
        process.WaitForExit();
        if (process.ExitCode == 0){ //OK
            Logger.Log($"Script␣was␣executed␣correctly,executing␣post␣
                execution␣actions␣.␣.␣.");
            script.PostExecutionActions();
        }
        else{
            Logger.Log($"Error␣occured␣during␣execution,executing␣post␣
                execution␣actions␣.␣.␣.");
            script.PostExecutionActions(false);
        }
    }
    catch (Exception ex){
        Logger.Log($"Error␣occured␣during␣execution,executing␣post␣
            execution␣actions␣.␣.␣.");
        script.PostExecutionActions(false);
    }
}
```

6 Architecture

6.1 MVVM

Design pattern MVVM (Model-View-ViewModel) is a software architecture patter designed with the goal of having a clear separation between the user interface (frontend) and the business logic (backend) [13]. There are different freameworks that use this standard, such as Angular, .NET WPF...

This application needs to be able to execute in CALCULA operating system (Linux), so it is necessary to use a framework that is compatible with this system. Originaly, it was going to be developed using .NET WPF, but it was decided to use Avalonia instead, as it is a cross-platform framework that can be executed in different operating systems. The differences between both frameworks are very low, mainly being the name of the files (.axaml instead of .xaml) and the way to define the user interface.

This architecture is divided in three main components:

- Model: It contains the data and most of the app logic. It structures the information (classes and identities), how to retrieve it and how to manage it (services, data bases and controllers).
- View: It defines all the elements in the user interface and what the user will see and will interact with. The only responsability of the code defines in this sections is to define the visual structure of the app and to create input elements for the user to interact with.
- ViewModel: It is the bridge between the Model and the View. It is the one that controls the flow of data between the Model and the View.

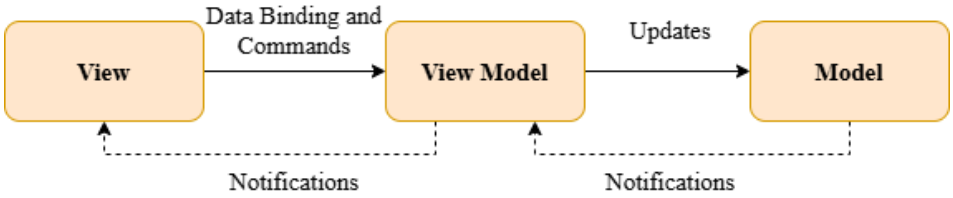

Figure 2: MVVM design pattern

The most important advantages of using this desing pattern are:

- Changes only affect the view or the model, not both
- Separated testing for views and models
- User interface can be modified without affecting the model
- Teams with different developers can work on the same project simultaneously without affecting each other

6.1.1 Observable Object

In order be able to update the user interface automatically when the data changes, it is used Data Binding, a feature of the MVVM pattern. It is implemeted using ObservableObject as base class.

```
public class User : ObservableObject
{
    private string name;

    public string Name
    {
        get => name;
        set => SetProperty(ref name, value);
    }
}
```

In the example, it can be seen how a variable is updated, when it is detected that the new value is different from the old one, SetProperty is called, which implements INotifyPropertyChanged interface, which is the one in charge to notify the corresponding view.

In order that the view is updated, it is necessary to call the variable with the same name and to indicate that it is Bindable in the corresponding .axaml file, as it can be seen in the following example:

```
<TextBlock Text="{Binding␣Name,␣Mode=TwoWay,␣UpdateSourceTrigger=
    PropertyChanged}" />
```

In this case, mode option indicate that variable can be updated from the view to the model and viceversa, and update source trigger option indicate that the update will be done when the property changes. Other options can be used, but these are the ones that are most common and that have been used in this project.

6.1.2 Relay Command

Relay command is used to implement commands in the MVVM pattern. It is implemeted using ICommand as base class. It can be used in different ways, but in this project it has been implemented using the following pattern for all View Models:

```
public class ViewModel : ObservableObject
{
  public ICommand Cmd => new RelayCommand(Execute, CanExecute);
  private void Execute(object _)
  {
      ExecutionActions();
  }

  private bool CanExecute(object _)
  {
      return true;
  }
}
```

In order to execute the command, it is necessary to link an interface element of the .axaml file to the Execute method

```
<Button Content="Execute" Command="{Binding␣Cmd}" />
```

6.2 SOLID principles

SOLID principles are a set of five principles that are used to design software with the obejective of making it easy to maintain and modify, making a project more maintainable ands scalable [16]. It lowers the dependency between classes, minimizing the impact of changes and making the code more reusable, mantainable, flexible and stable.

They also support growth in development, making it easier to add new features without the need of modifying already implemented code.

These principles are:

- Single Responsibility Principle (SRP)
- Open/Closed Principle (OCP)
- Liskov Substitution Principle (LSP)
- Interface Segregation Principle (ISP)
- Dependency Inversion Principle (DIP)

6.2.1 Single Responsibility Principle

Statets that *a class should have only one reason to change*, meaning that it should have only one responsibility, job or purpose within the softwar project.

In order to apply this principle, it is necessary to divide the code into different classes, each one with a specific responsibility. It is important to note that this principle is not always possible to apply, but it is a good practice to try to do so.

6.2.2 Open/Closed Principle

States that *an entity should be open for extension, but closed for modification*, meaninig that it should be easy to extend the functionality of the entity without modifying its existing code.

In order to apply this principle it can be used abstraction by implementing Interfaces and Abstract classes and by using different design patterns like Strategy or Decorator.

6.2.3 Liskov Substitution Principle

Introduced in 1987 by Barbara Liskov, it states that *derived or child classes must be substitutable for their base or parent classes*, ensuring that any class can be used in place of its parent class without any loss of functionality.

In order to apply this principle, it is important to ensure that subclasses can be used anywhere where the base class is used without changing the expected behaviour, avoiding

overriding methods and making sure to not thow NotImplementedException. This can be done by using Interfaces and Abstract classes.

6.2.4 Interface Segregation Principle

This principle applies directly to interfaces, and it states that *do not force any client to implement an interface which is irrelevant to them*. The main objective of this principle is to not implement too big interfaces and to implement instead smaller and more specific interfaces.

If this principle is used in a project, the corresponding software architecture should have many interfaces implementations with specific purposes.

6.2.5 Dependency Inversion Principle

States that *high-level modules should not depend on low-level modules, but both should depend on abstractions*, ensuring that the dependencies are inverted and that the high-level modules are not dependent on the low-level modules.

In order to apply this principle, it is suggested to use abstractions and interfaces to define the dependencies between modules.

6.3 Project architecture

Applying the MVVM design pattern and the SOLID principles, the project architecutre can be summed up as follows:

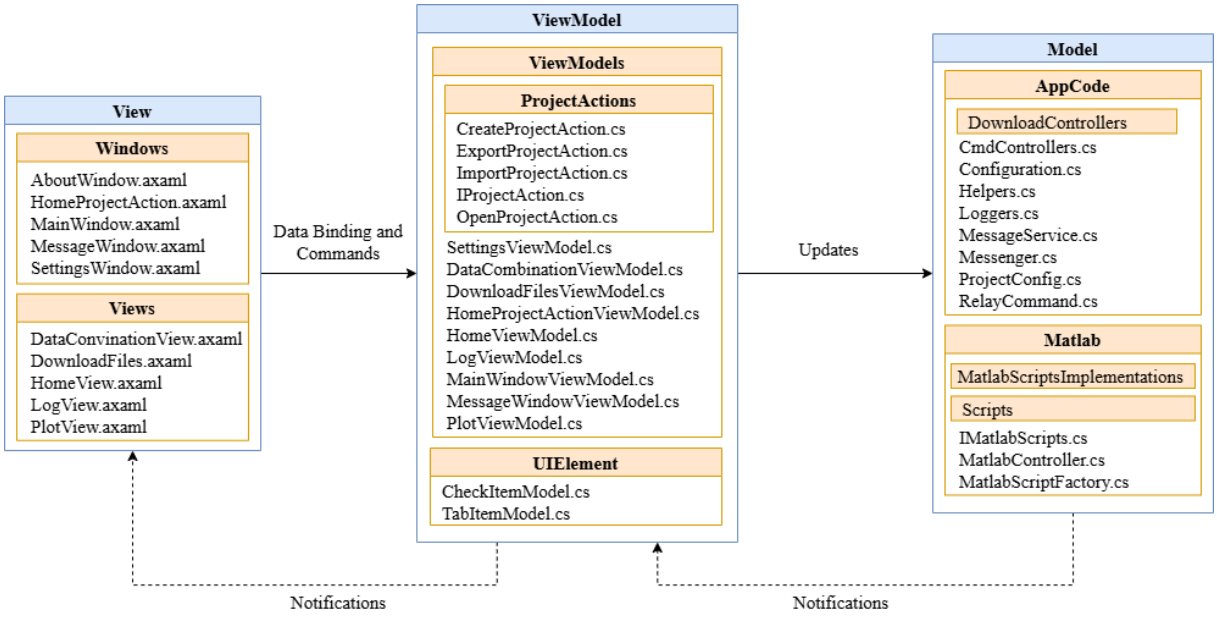

Figure 3: MVVM design pattern applied to the project architecure

In Figure 3 it can be seen how the Views are divided into different folders. The Windows folder contains all the files containing the definitions of the different windows and dialogs

of the application, while the Views folder contains all the files containing the definitions of different UserControls. UserControls are used as Content in the MainWindow, in this case, as Contents of the different tab items.

The ViewModel section contains all the files containing the definitions of the different ViewModels of the application. Most of them are located in the ViewModels folder, but it can also be appreciated that there is a folder named UIElement, where the models for CheckItems and TabItems are located. With this classes, adding these items to Views and ViewModels is easier to mantain and update.

Finally, the Model component is composed by all the internal classes of the application. It has been divided by AppCode, where is located all the implementations related to app Logic, and by Matlab, where all the MATLAB scripts are located aswell as the necessary implementations for the app to execute them.

7 Graphic User Interface (GUI) design

8 Implementation

8.1 Set up development enviroment

9 Testing

Table 2: 1st round testing - Project workspace management

#	Test	Expected result	OK/KO	Comments
1	1. Click icon for creating new project 2. Choose a directory 3. Choose a name 4. Click create	Project is created with defined name and directory, home view is closed and "download files", "Data convination" and "Plotting" views are available.	OK	None
2			OK	None
3			OK	None
4			OK	None
5			OK	None
6			OK	None
7			OK	None
8			OK	None
9			OK	None
10			OK	None
11			OK	None
12			OK	None
13			OK	None
14			OK	None
15			OK	None
16			OK	None
17			OK	None
18			OK	None
19			OK	None
20			OK	None
21			OK	None
22			OK	None
23			OK	None
24			OK	None
25			OK	None
26			OK	None
27			OK	None
28			OK	None
29			OK	None
30			OK	None

10 Results

This should include your data analysis and findings

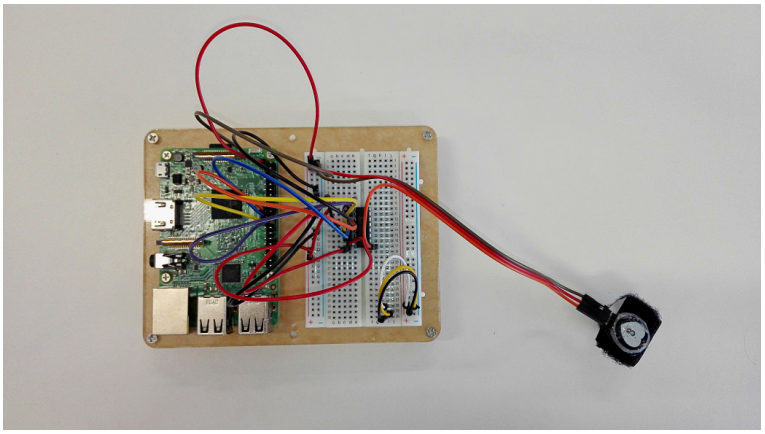

Figure 4: Prototype setup.

Table 3: This is the other caption. Since the trial size of the experiments showed is one second, the number of *Target* and *Impostor* data corresponds to number of trials or seconds

Dataset	Label	Train	Validation	Develop	Test
First	Target	135	45	30	30
	Impostor	5,220	1,740	1,890	2,880
	#Subjects		31		12
Second	Target	144	80	48	48
	Impostor	2,014	1,119	1,343	1,545
	#Subjects		15		5

Algorithm 1 Temperature-Distributed algorithm

procedure TEMP-SPREAD($GN_i, HN_j, temperatures$) ▷ Lowest temperature priority
  $temperature\_list \leftarrow short(temperatures)$
  $max_temperature \leftarrow max(temperature_list)$
  $ThresHold \leftarrow 0.5$
  $temperature\_impact \leftarrow 0.2$
  for GN_i in $i = 1, 8$ **do** ▷ Iterate every hardware node on the given GN
    $it\_temperature \leftarrow temperature\_list(GN_i)$
    $temp\_weight \leftarrow \frac{max\_temperature - it\_temperature}{max\_temperature} * temperature\_impact$
    $\omega(Master - GN_i) \leftarrow ThresHold * temp\_weight$
    for HN_j in $j = 1, n$ **do**
      if $available\_accel_{i,j} > busy\_accel_{i,j}$ **then**
        $policy_\omega = \frac{AvailableHW}{TotalHW} * ThresHold$
        $\omega(GN_i - HN_{i,j}) \leftarrow ThresHold + policy_\omega$
      else
        $\omega(GN_i - HN_{i,j}) \leftarrow 1$
  $node \leftarrow find\_djistra\_shortest\_path(Master\_Node, aux\_node)$
  $returnnode\ b$ ▷ The gcd is b

11 Conclusions

References

[1] Wolfgang Skala. Drawing gantt charts in latex with tikz.

[2] Albert Einstein. Zur Elektrodynamik bewegter Körper. (German) [On the electrodynamics of moving bodies]. *Annalen der Physik*, 322(10):891–921, 1905.

[3] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1993.

[4] Donald Knuth. Knuth: Computers and typesetting.

[5] C.N.R. IMAA. Actris-earlinet data portal. Web Application available at `https://data.earlinet.org/earlinet/desktop.zul`, 2024.

[6] NASA. Aerosol robotic network (aeronet). Web Application available at `https://data.earlinet.org/earlinet/desktop.zul`, 2025.

[7] Miriam Martínez Canelo. ¿qué es la programación orientada a objetos? Web Application available at `https://profile.es/blog/que-es-la-programacion-orientada-a-objetos/`, 2025.

[8] MathWorks. Matlab. Web Application available at `https://es.mathworks.com/products/matlab.html`, 2025.

[9] C.N.R. IMAA. Actris-earlinet rest api. Web Application available at `https://data.earlinet.org/api/swagger-ui/`, 2025.

[10] NASA. Aerosol optical depth. Web Application available at `https://aeronet.gsfc.nasa.gov/print_web_data_help_v3_new.html`, 2025.

[11] NASA. Raw products aerosol optical depth. Web Application available at `https://aeronet.gsfc.nasa.gov/print_web_data_help_v3_raw_sky_new.html`, 2025.

[12] NASA. Aerosol inversions. Web Application available at `https://aeronet.gsfc.nasa.gov/print_web_data_help_v3_inv_new.html`, 2025.

[13] Microsoft. Mvvm. Web Application available at `https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm`, 2025.

[14] Microsoft. Observableobject. Web Application available at `https://learn.microsoft.com/en-us/dotnet/communitytoolkit/mvvm/observableobject`, 2025.

[15] Microsoft. Relaycommand. Web Application available at `https://learn.microsoft.com/en-us/dotnet/communitytoolkit/mvvm/relaycommand`, 2025.

[16] GeeksforGeeks. Solid principles. Web Application available at `https://www.geeksforgeeks.org/system-design/solid-principle-in-programming-understand-with-real-life-examples/`, 2025.

Appendices

Appendices may be included in your thesis but it is not a requirement.