Welcome to My Portfolio

A Humble Collection of My Projects

Done by : AYAT ACHRAF

Portfolio Projects Summary

❑ ***Control Systems***

•**Speed Regulation of a DC Motor**: Developed a control system to regulate the speed of a DC motor.

•**Boost Converter**: Implemented and analyzed a DC-DC boost converter for voltage regulation applications.

❑ ***Arduino***

•**Line Following and Obstacle Avoidance Robot**: Built and programmed a robot to autonomously follow a line and avoid obstacles.

❑ ***ESP32***

•**Plant Watering System**: Created an automated system to monitor soil moisture and water plants as needed using an ESP32 microcontroller.

❑ ***Model-Based Design (MBD)***

•**Wiper System Project**: Designed and simulated an automotive wiper control system using MBD principles.

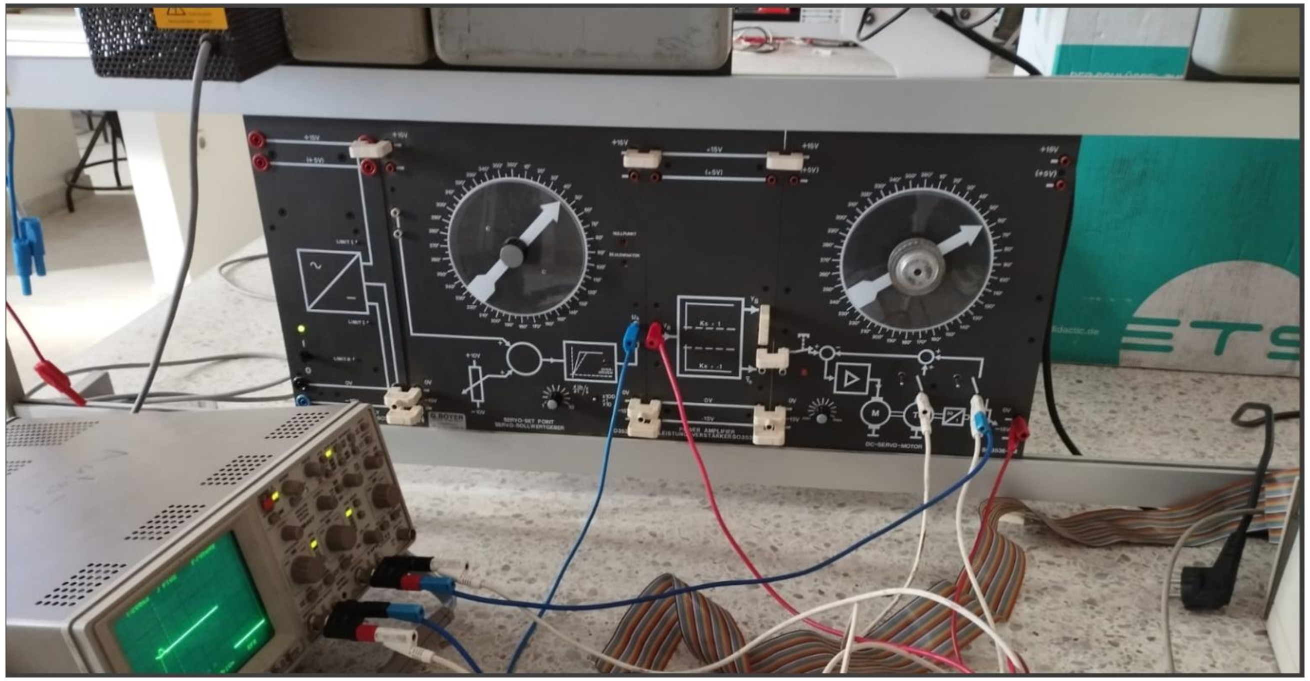
❑ ***Electronics***

•**Rectifier**: Implemented a rectifier circuit for AC to DC conversion applications.

Introduction:

The primary goal of this application is to determine the transfer function of a real DC motor to control its speed using state feedback, also known as the pole placement method.

The process began by setting up the DC motor, connecting all inputs, outputs, and the power source to an oscilloscope through a Cassy board interface. An open-loop analysis was first conducted to obtain the DC motor’s transfer function. Finally, a closed-loop study using pole placement was performed to increase the system's degrees of freedom, enhancing stability, response time, and steady-state error.



Practical Section (Part 1)

We generally have a position sensor placed directly on the axis and a speed sensor. These sensors provide voltages u_θ and u_w proportional to the measured quantities, which can therefore be written as: $\boldsymbol{u_\theta = P.\theta(t) et\, u_w = T.\dot{\theta}(t)}$
Because DC motors provide precise speed control by adjusting the armature voltage.

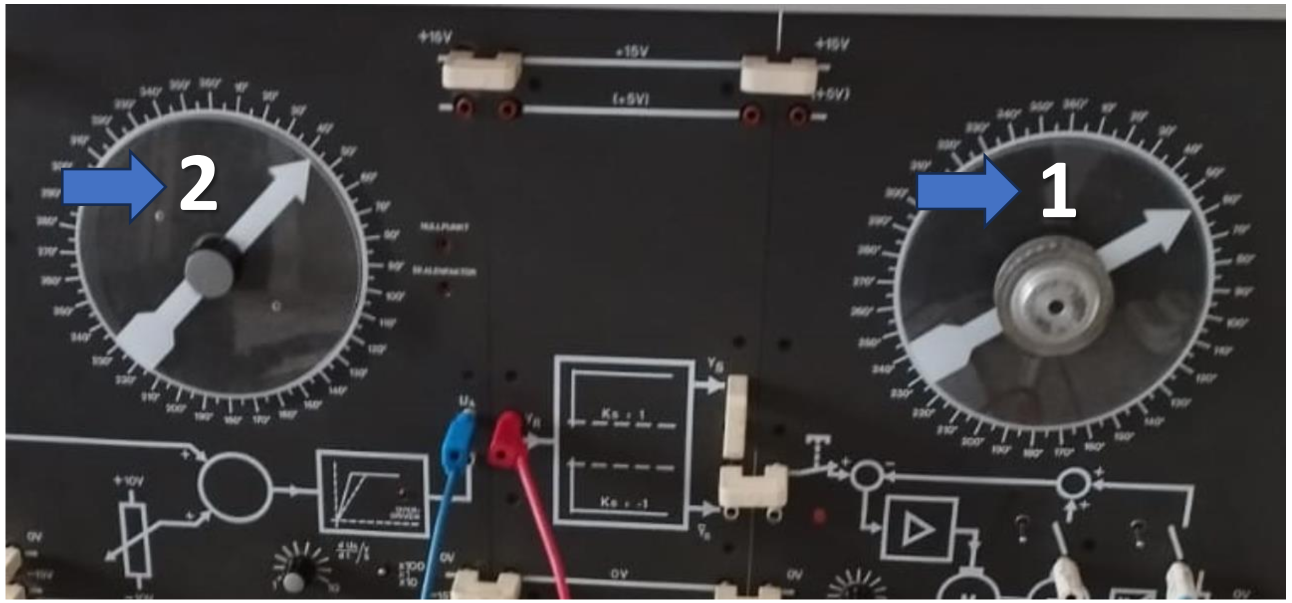
- Determining the coefficients P and T:

For P, we chose the position, and each one corresponds to a voltage:

$$\theta_1 = 30 \leftrightarrow u_{\theta_1} = 1.8\,V$$

$$\theta_2 = 50 \leftrightarrow u_{\theta_2} = 2.37\,V$$

$$P = \frac{\Delta u_\theta}{\Delta \theta} = \frac{2.37 - 1.8}{50 - 30} = 2.85 * \ 10^{-2} V/° \approx 0.03\,V/°$$



1 ➡ ***Indicator for Input Position***

2 ➡ ***Indicator for Output Position***

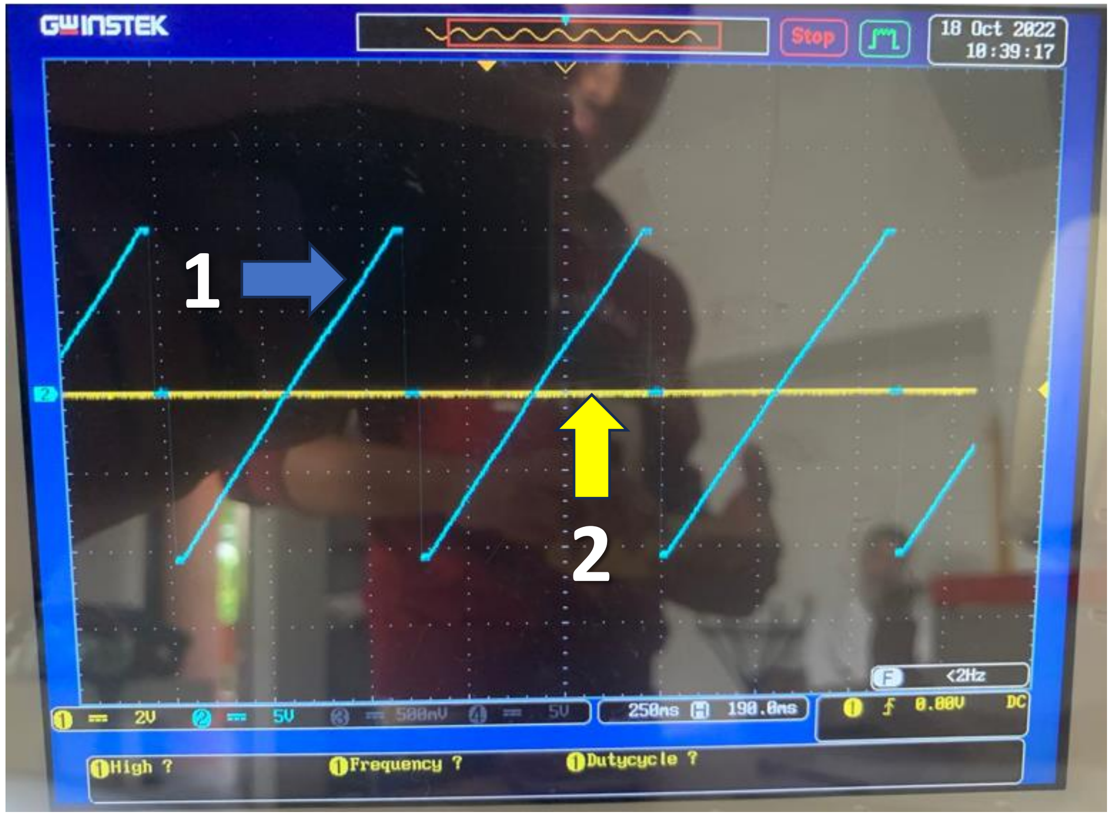
For T, we chose the speed, and each one corresponds to a voltage.

Note: There are two methods to find the speed.
Method 1: Calculating the speed using 3 periods from the oscilloscope.
Method 2: Using a monitor and counting the number of revolutions during one minute.

$$\dot{\theta}_1 = 83\ tr/min \leftrightarrow u_w = 2.4\ V$$

$$\dot{\theta}_1 = 75\ tr/min \leftrightarrow u_w = 2.2\ V$$

$$T = \frac{\Delta u_w}{\Delta \dot{\theta}} = \frac{2.4 - 2.2}{83 - 75} = 25 * \ 10^{-3}\ tr/min \approx 0.025\ tr/min$$

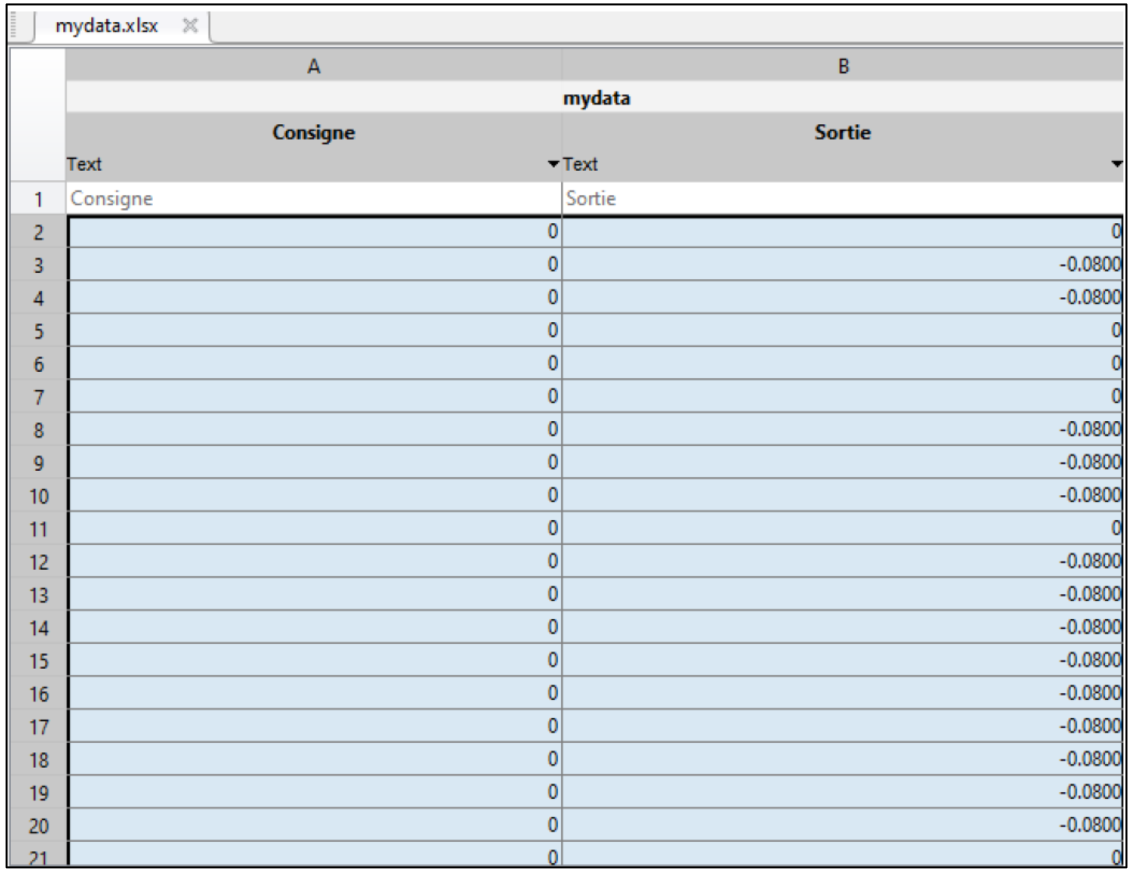


1 Speed voltage $\boldsymbol{u_w}$ (Ramp)

2 Position voltage $\boldsymbol{u_\theta}$ (Step)

- ***Simulation***

In this section, we used a digital oscilloscope that enables us to export an Excel sheet (see the file in the resource section) and import it into Simulink to begin the identification process.
The data imported into MATLAB corresponds to the setpoint (Step) and the output (voltage speed) of the system



mydata.xlsx

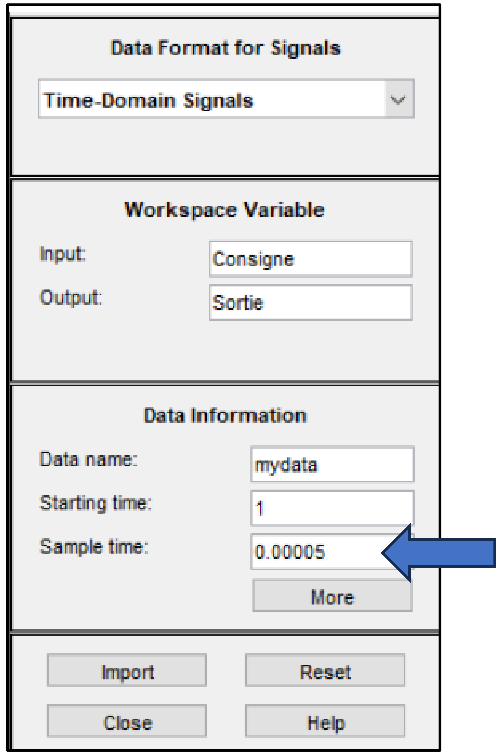
	A	B
	mydata	
	Consigne	Sortie
	Text	Text
1	Consigne	Sortie
2	0	0
3	0	-0.0800
4	0	-0.0800
5	0	0
6	0	0
7	0	0
8	0	-0.0800
9	0	-0.0800
10	0	-0.0800
11	0	0
12	0	-0.0800
13	0	-0.0800
14	0	-0.0800
15	0	-0.0800
16	0	-0.0800
17	0	-0.0800
18	0	-0.0800
19	0	-0.0800
20	0	-0.0800
21	0	0

illustration of the data imported into MATLAB

- ***Identification***

For identification,we used the Ident function in Simulink to perform system identification our linear plant, which involves estimating the mathematical mdel of a dynamic system based on measured input-output data.

To find the suitable model by conducting trials, meaning selecting a transfer function, for example, with poles and without zeros, a function with delay, etc.


Data Format for Signals

Time-Domain Signals

Workspace Variable

Input: Consigne

Output: Sortie

Data Information

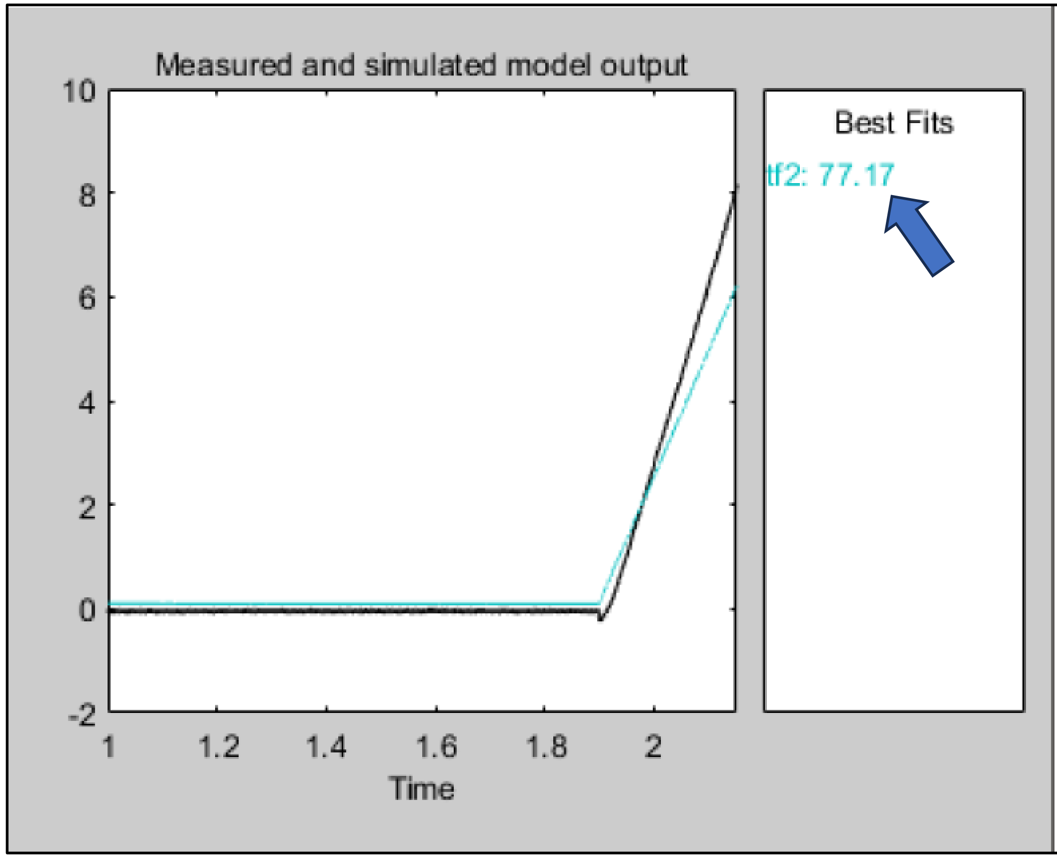
Data name: mydata

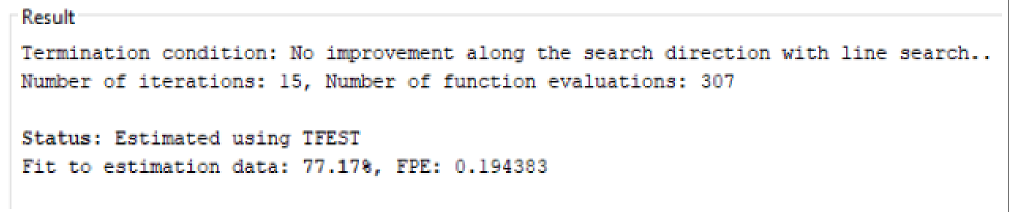
Starting time: 1

Sample time: 0.00005

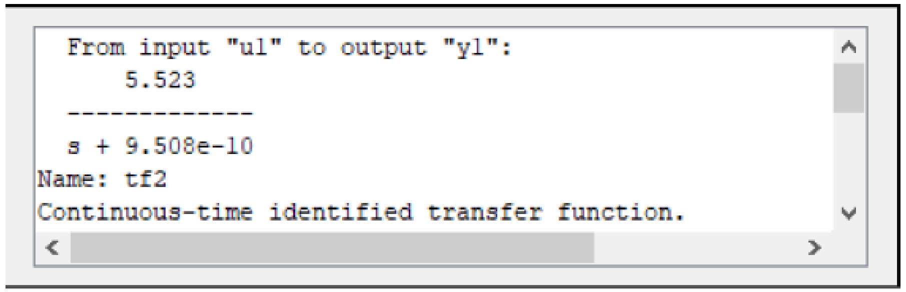
Determination of Sample time

- ***case of a function with poles and without zeros***





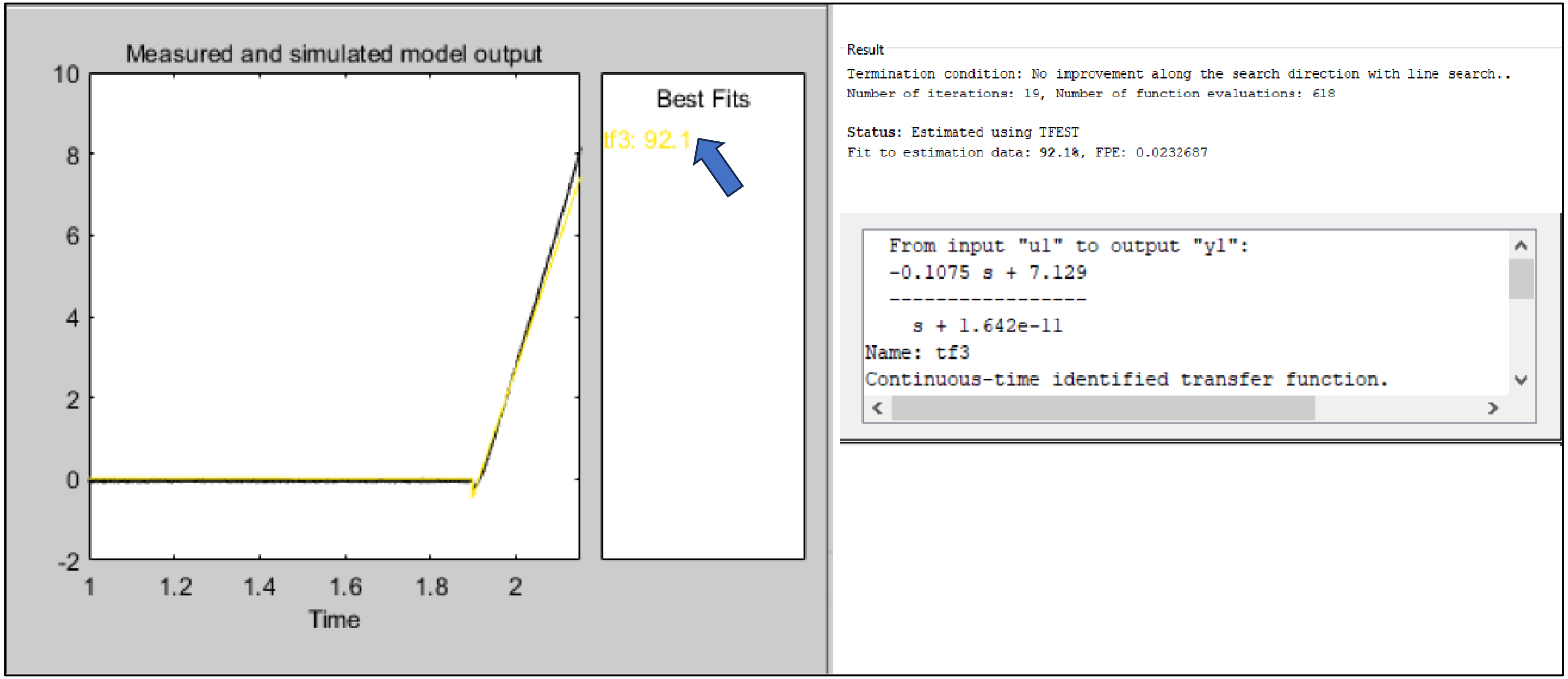
```
Result
Termination condition: No improvement along the search direction with line search..
Number of iterations: 15, Number of function evaluations: 307

Status: Estimated using TFEST
Fit to estimation data: 77.17%, FPE: 0.194383
```



```
  From input "u1" to output "y1":
      5.523
  -------------
  s + 9.508e-10
Name: tf2
Continuous-time identified transfer function.
```

- Estimated fit was 77.17%

- ***case of a function with poles and zeros***

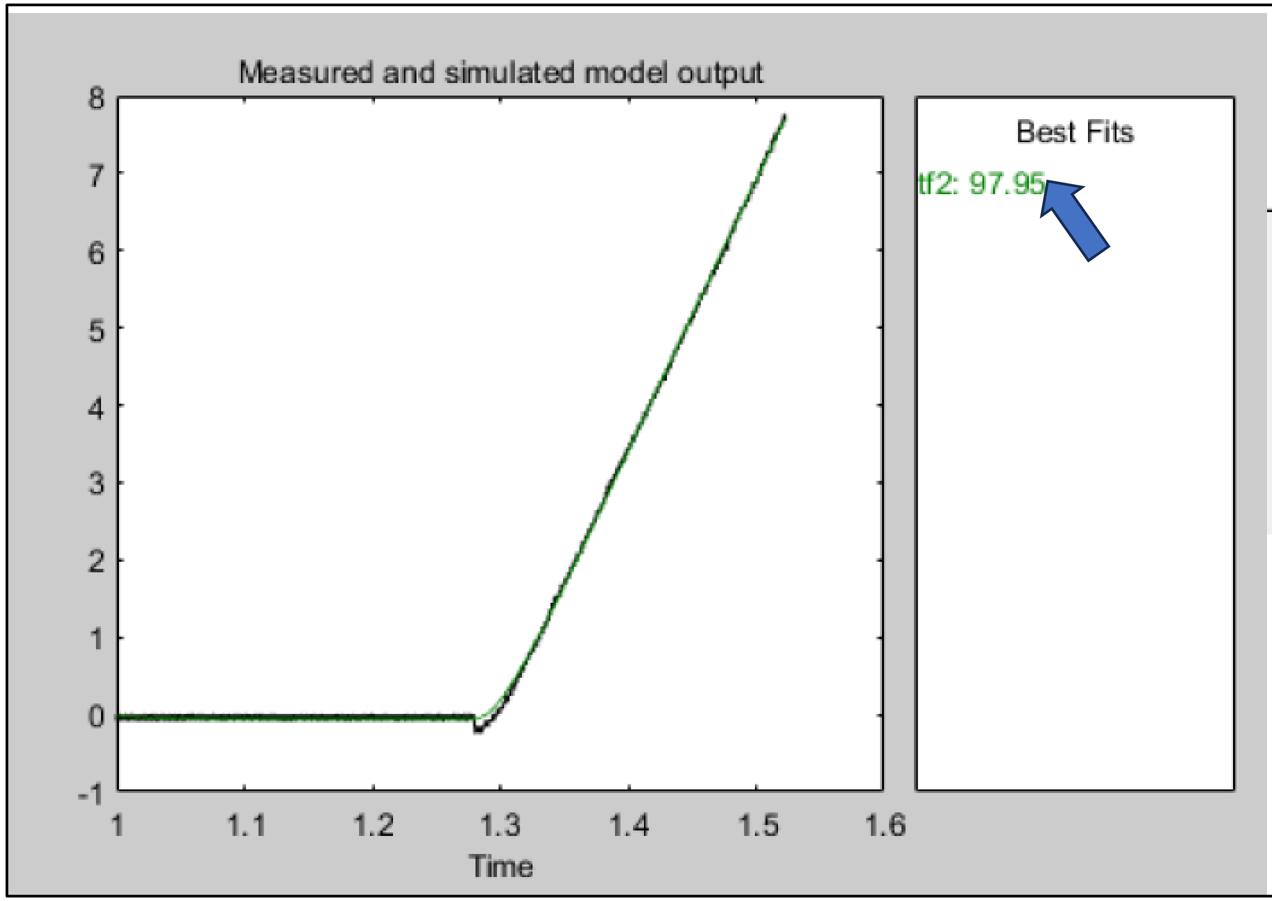


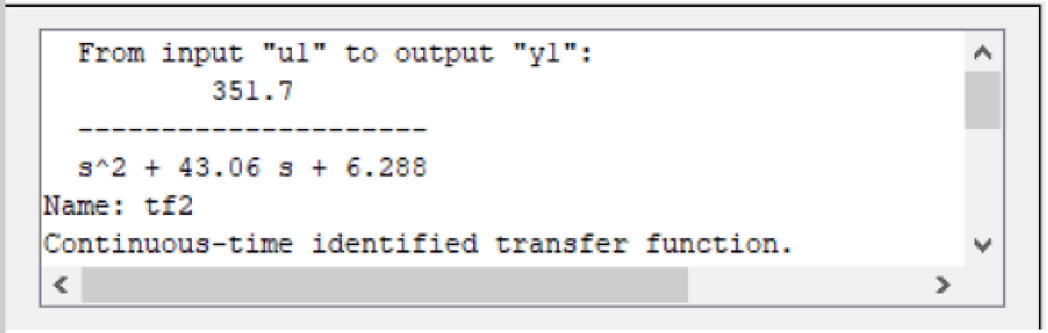
```
Result
Termination condition: No improvement along the search direction with line search..
Number of iterations: 19, Number of function evaluations: 618

Status: Estimated using TFEST
Fit to estimation data: 92.1%, FPE: 0.0232687
```

```
  From input "u1" to output "y1":
  -0.1075 s + 7.129
  -----------------
    s + 1.642e-11
Name: tf3
Continuous-time identified transfer function.
```

- Estimated fit was 92.1%

- ***case of a function with 2 poles and without zeros***



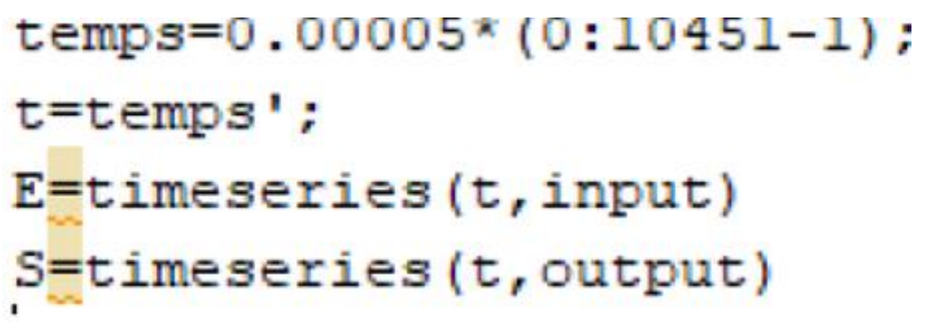


```
  From input "u1" to output "y1":
           351.7
  ---------------------
  s^2 + 43.06 s + 6.288
Name: tf2
Continuous-time identified transfer function.
```

Based on these trials, we observe that the transfer function with two poles and without zeros is the closest or most suitable for our system, as the fit is **97.67** %

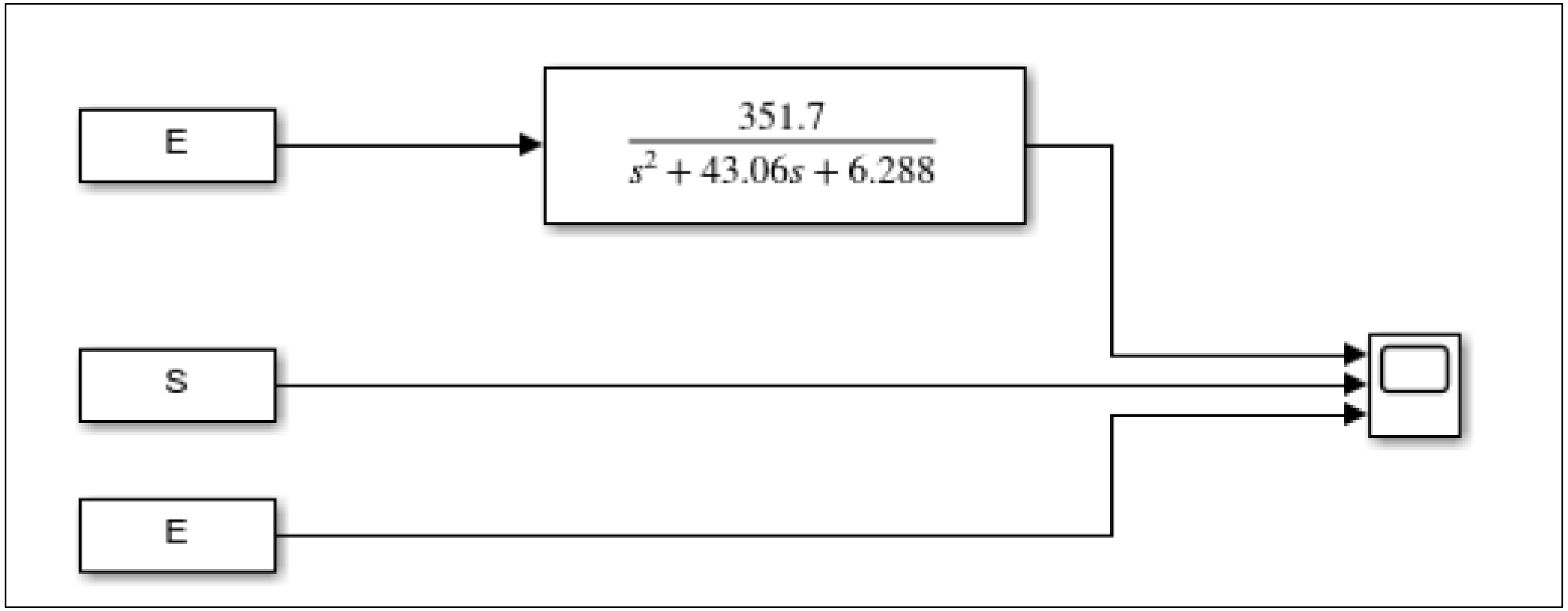
Comparison of the system response with the output data in Simulink

To visualize the actual curve and the corresponding model curve, we use the **Timeseries function** to represent the evolution of either the input or output over time.



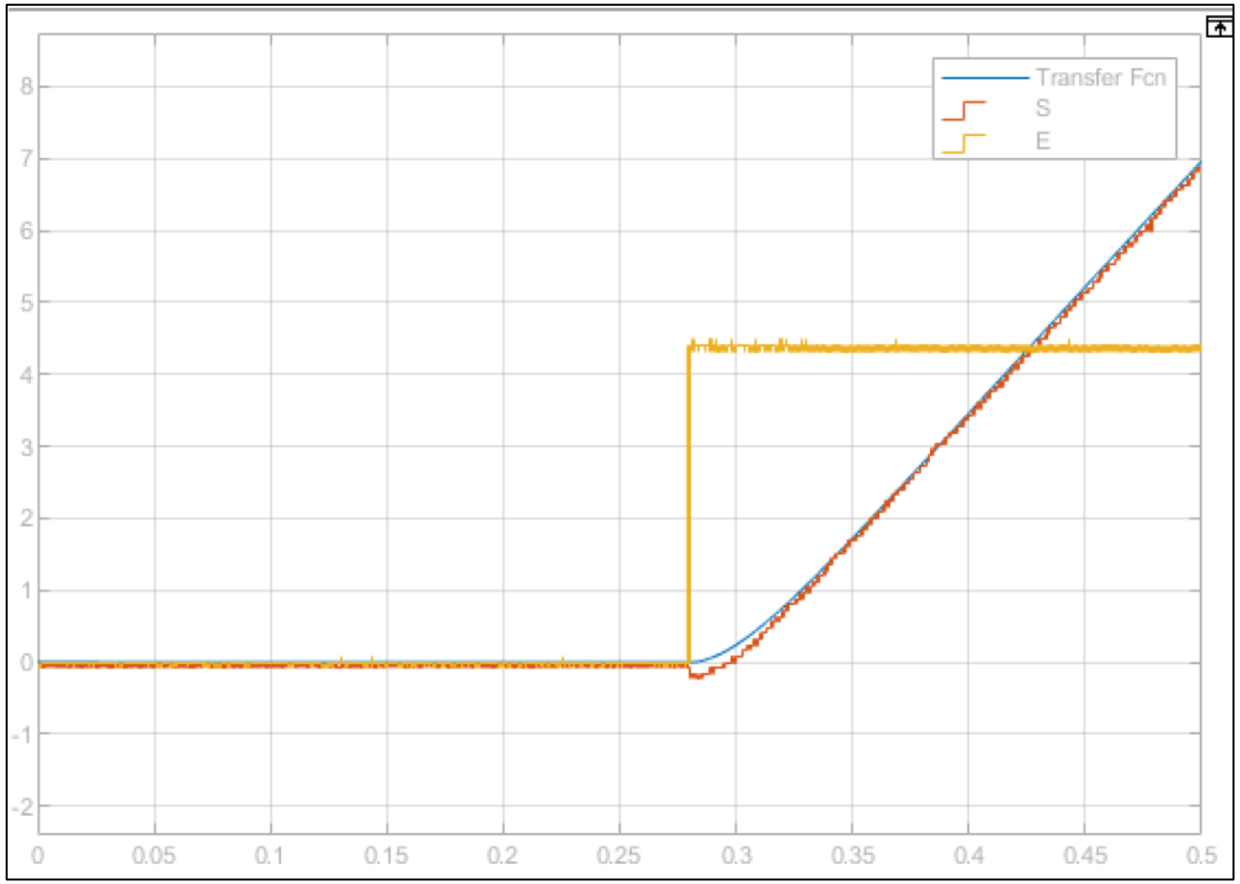
```
temps=0.00005*(0:10451-1);
t=temps';
E=timeseries(t,input)
S=timeseries(t,output)
```

- 0.00005 stand for Sample time.
- 10451 represents the length of the Excel sheet imported from the oscilloscope.
- E represents the Input.
- S represents the output.

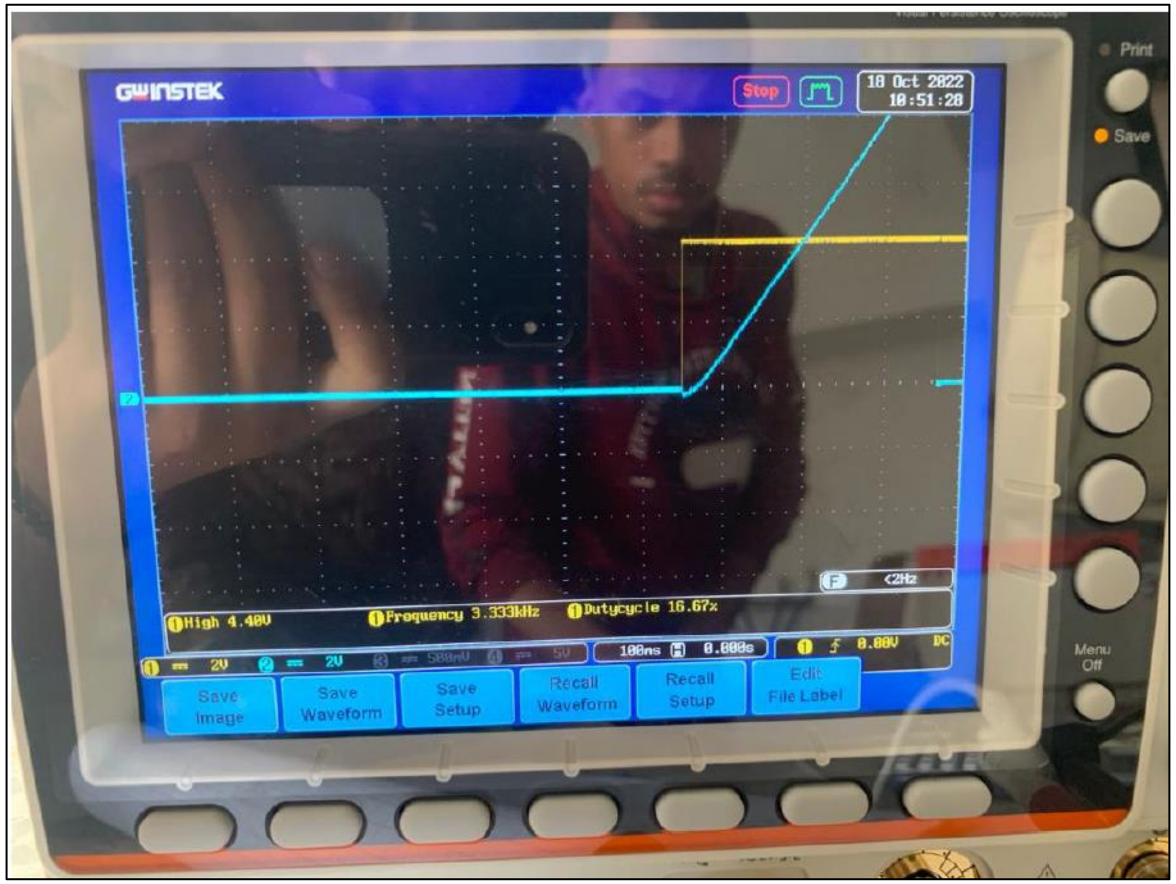


Simulation of the transfer function in open loop

Result



Theoretical result

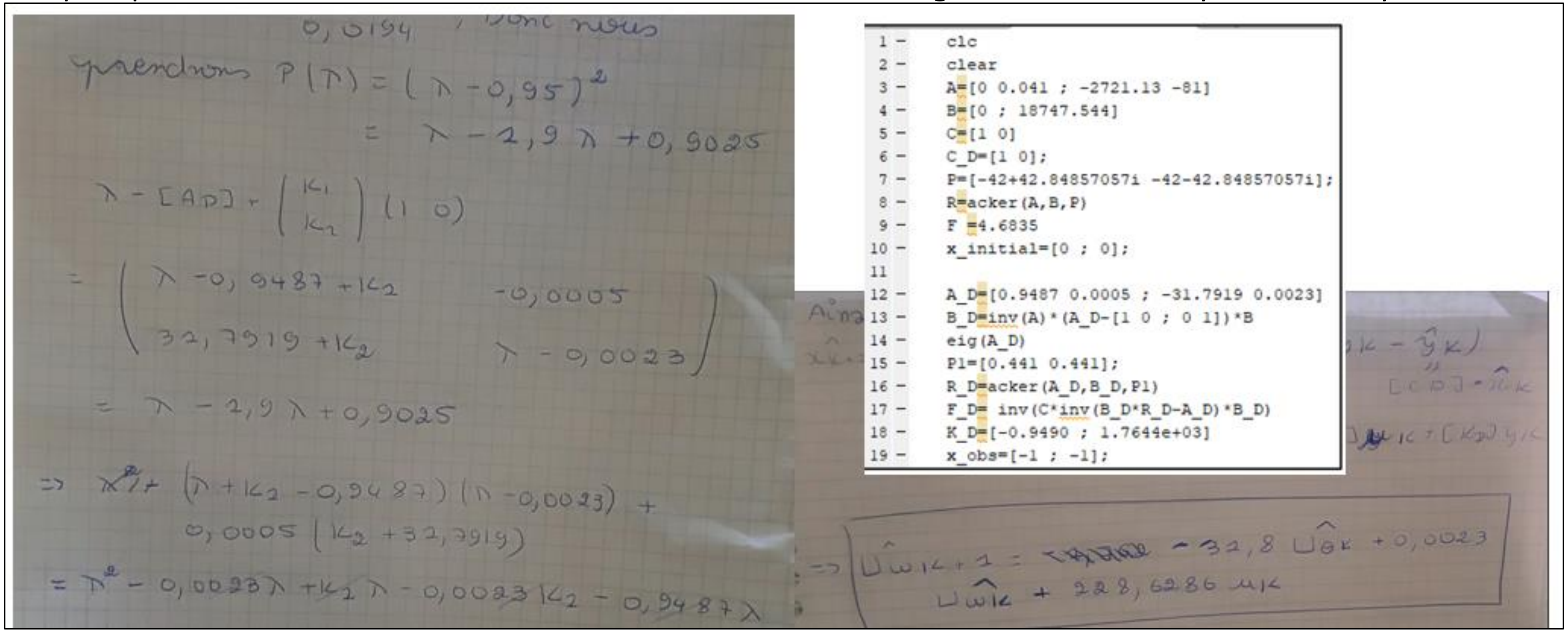


Experimental result

Conclusion : The identification of the DC motor through the transfer function aligns closely with the actual results.

Practical Section (Part 2)

After identifying our system, it is evident that it is not stable because the output (speed) does not converge. Therefore, we will use the pole placement method to address the issue based on linear algebra and control systems theory.



0, 0194 , Donc nous

prenons $P(\lambda) = (\lambda - 0,95)^2$

$= \lambda - 2,9\lambda + 0,9025$

$\lambda - [A_D] + \begin{pmatrix} K_1 \\ K_2 \end{pmatrix} (1 \quad 0)$

$= \begin{pmatrix} \lambda - 0,9487 + K_2 & -0,0005 \\ 32,7919 + K_2 & \lambda - 0,0023 \end{pmatrix}$

$= \lambda - 2,9\lambda + 0,9025$

$\Rightarrow \lambda^2 + (\lambda + K_2 - 0,9487)(\lambda - 0,0023) +$
$0,0005 (K_2 + 32,7919)$

$= \lambda^2 - 0,0023\lambda + K_2\lambda - 0,0023 K_2 - 0,9487\lambda$

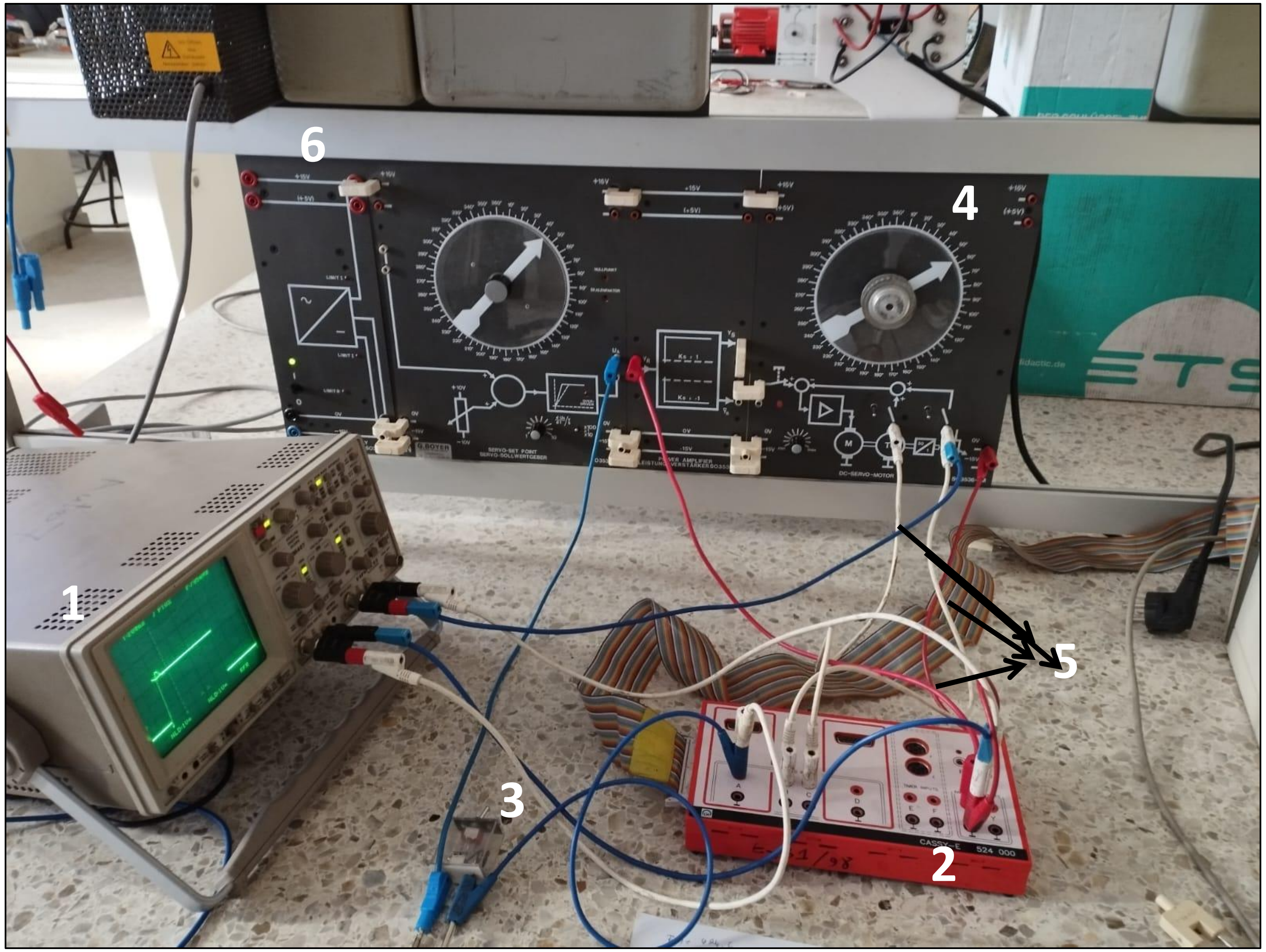
```
clc
clear
A=[0 0.041 ; -2721.13 -81]
B=[0 ; 18747.544]
C=[1 0]
C_D=[1 0];
P=[-42+42.84857057i -42-42.84857057i];
R=acker(A,B,P)
F =4.6835
x_initial=[0 ; 0];

A_D=[0.9487 0.0005 ; -31.7919 0.0023]
B_D=inv(A)*(A_D-[1 0 ; 0 1])*B
eig(A_D)
P1=[0.441 0.441];
R_D=acker(A_D,B_D,P1)
F_D= inv(C*inv(B_D*R_D-A_D)*B_D)
K_D=[-0.9490 ; 1.7644e+03]
x_obs=[-1 ; -1];
```

$\Rightarrow \boxed{\hat{\omega}_{k+1} = ~~[illegible]~~ - 32,8\,\hat{\theta}_k + 0,0023\,\hat{\omega}_k + 228,6286\,u_k}$

Extract from the theoretical calculation using the pole placement technique

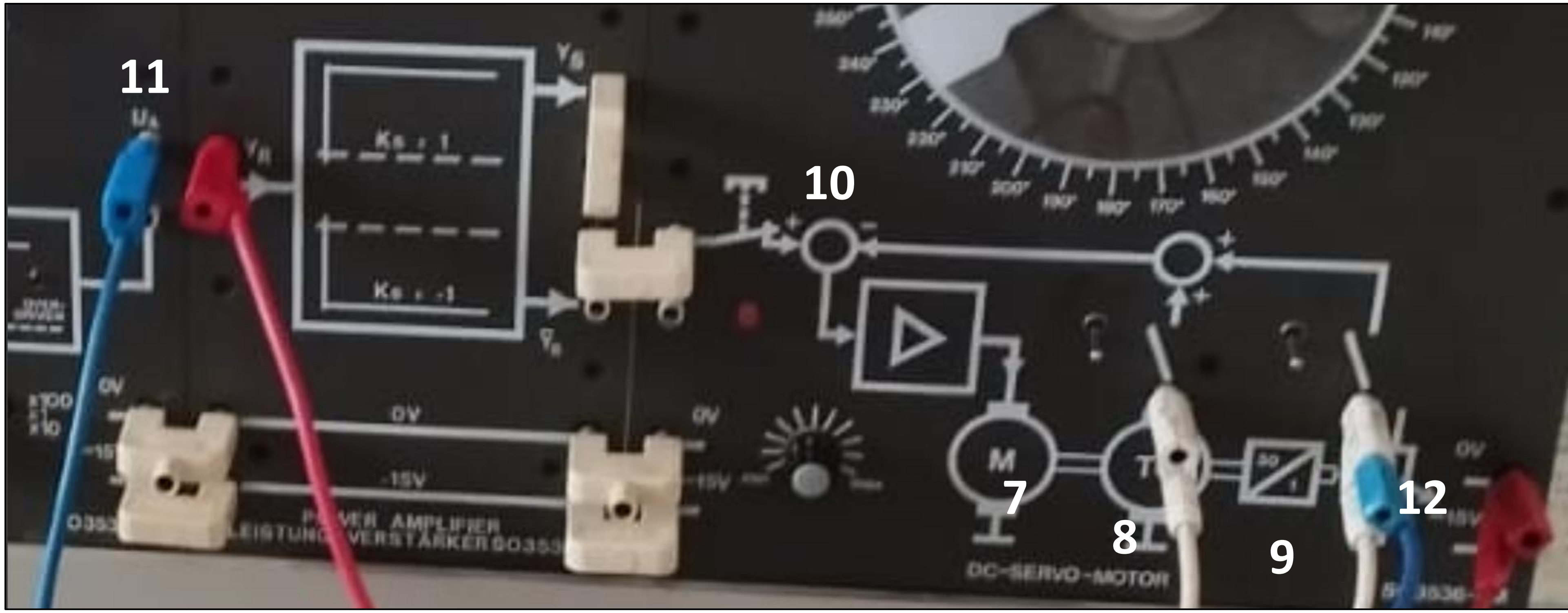
Closed Loop Experiment Overview:

1-Analog Oscilloscope
2-Data acquisition board
3-Switch
4-Experimental DC board
5-connection wires
6-Power supply

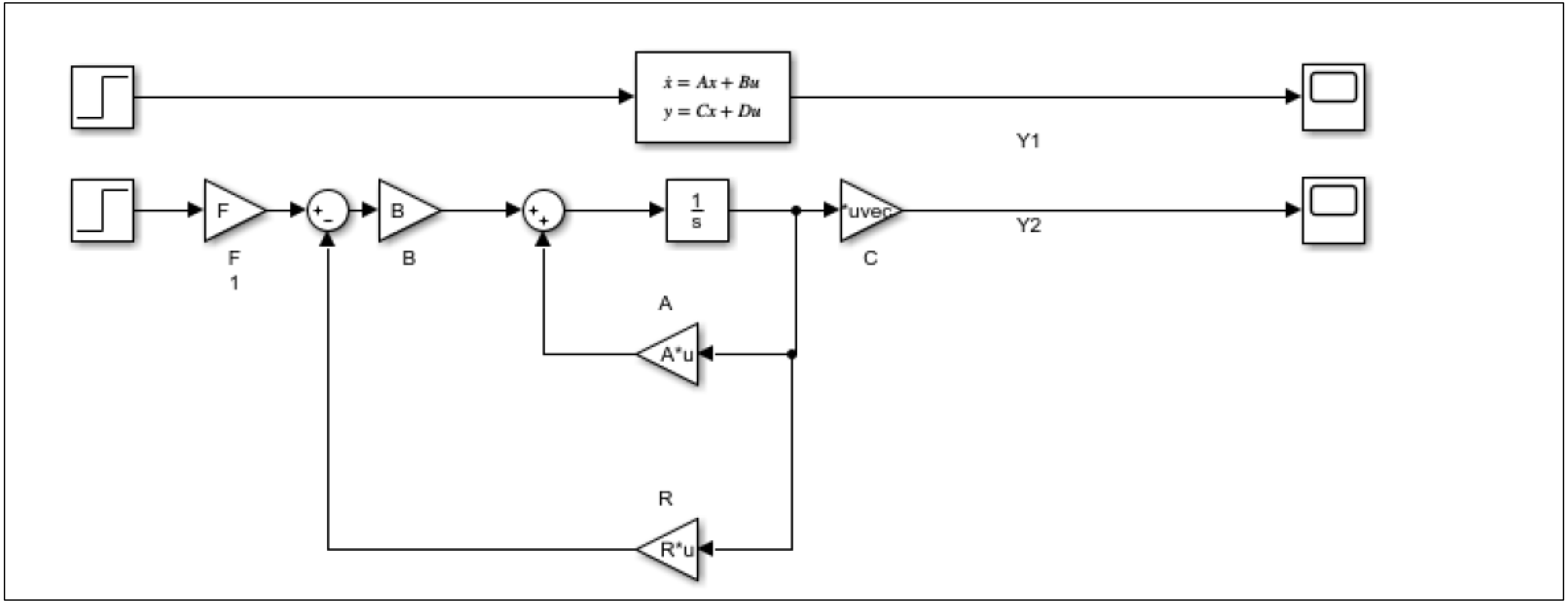


Closed Loop Experiment Overview:

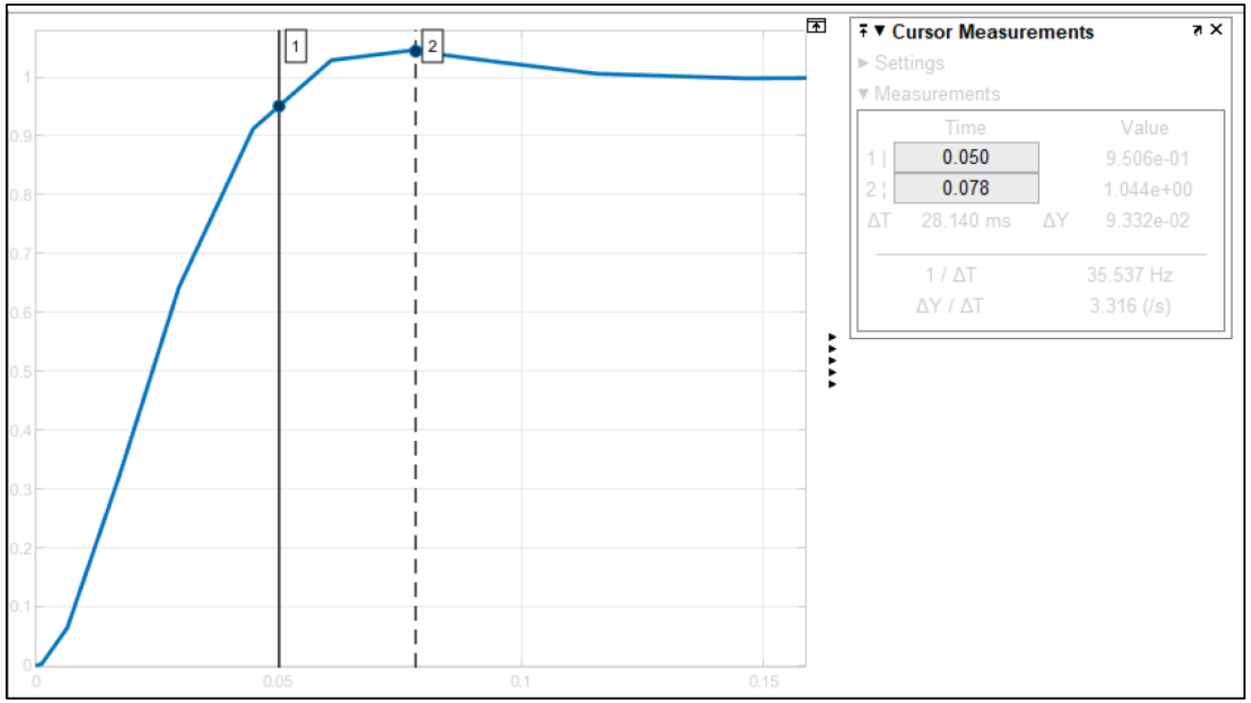
7-DC motor (Plant)
8-Speed sensor (Tachometer)
9-ADC
10-comparaison error
11-Input
12-Output



After performing the calculations, we conducted a closed-loop simulation using the pole placement method.



Result



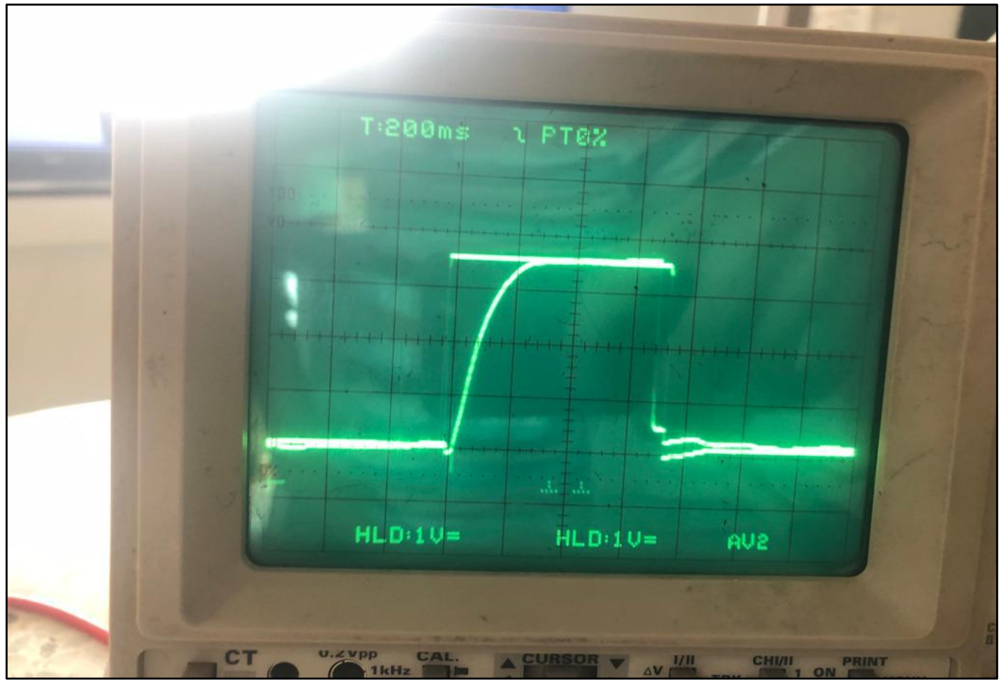
Cursor Measurements

▸ Settings

▾ Measurements

	Time		Value
1	0.050		9.506e-01
2	0.078		1.044e+00
ΔT	28.140 ms	ΔY	9.332e-02
1 / ΔT			35.537 Hz
ΔY / ΔT			3.316 (/s)

Theoretical result



Experimental result

Interpretation:

According to the simulation, it can be observed that the state feedback control meets the given specifications

Conclusion

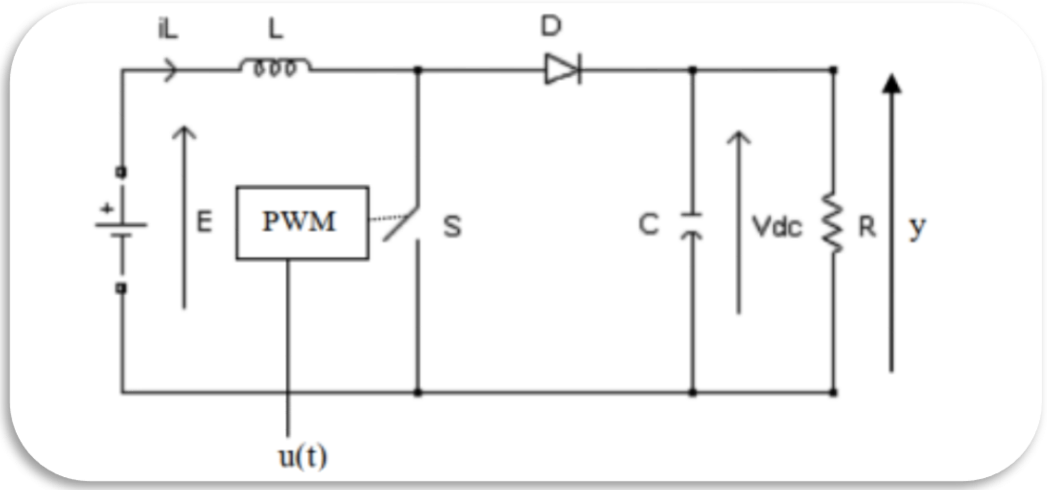
This project enabled me to utilize various blocks in Simulink and the Matlab workspace, as well as to deepen my understanding of control theory and carry out speed regulation for a DC motor.

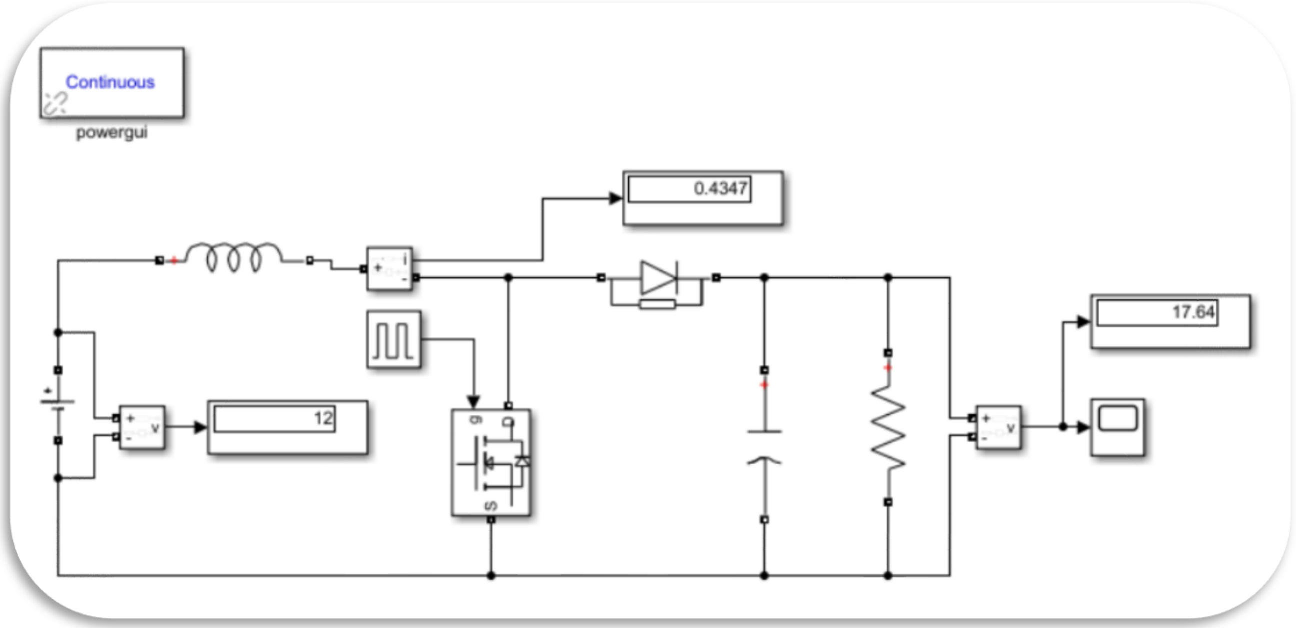
Note: The experimental results in the closed-loop case were obtained by implementing the calculated equations from the pole placement method on a computer using the Pascal language, which was also connected to the Cassy interface board.

Link to the original file: https://drive.google.com/file/d/1h17srxJ-sQ-dWpvwjYD8EoL-gjovfTw6/view?usp=drive_link

Abstract

- *This work allowed us to simulate and understand the operation of the boost converter (step-up or voltage-increasing chopper), which has many applications in our daily lives*





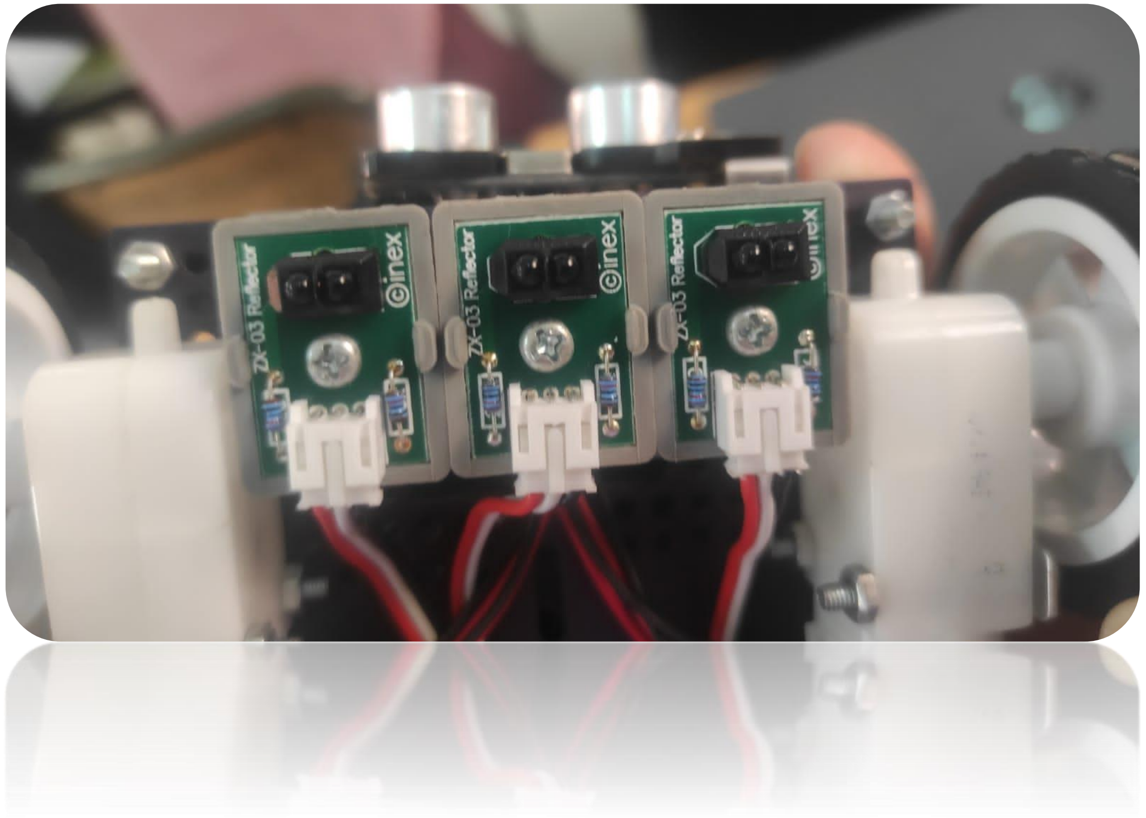
- The control of this circuit involved state feedback control (pole placement) around an operating point because its characteristic is nonlinear, so we worked within a linearity range to regulate the DC output voltage of the Boost Converter.

Link to the file: https://drive.google.com/file/d/1UbKRNH_-lUZpiVEorxEhlOV9ykBkSbkA/view?usp=sharing

Line-following robot and obstacle avoidance

This application was designed to program a four-wheeled robot for obstacle avoidance and line-following tasks, utilizing an ultrasonic sensor and infrared sensor for detection, an LCD display to show distance, and an Arduino ATX2 for control.

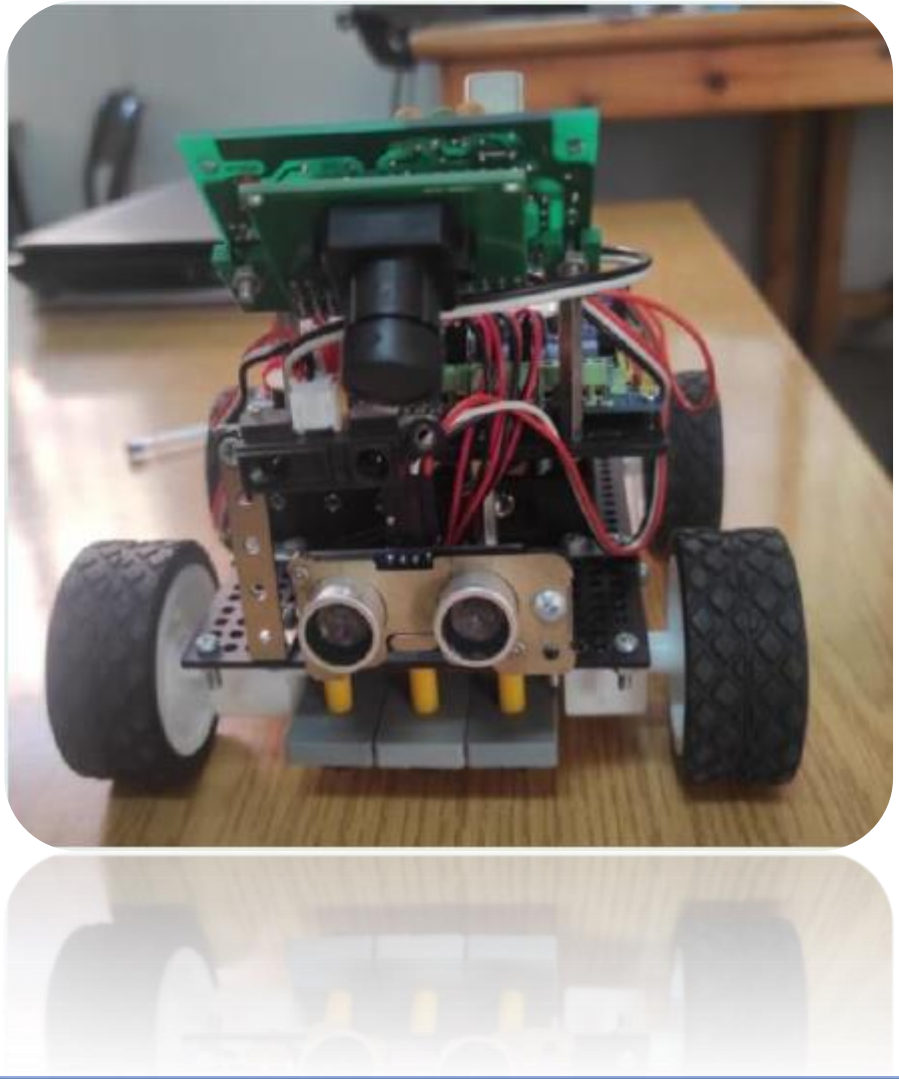
Infrared Sensor



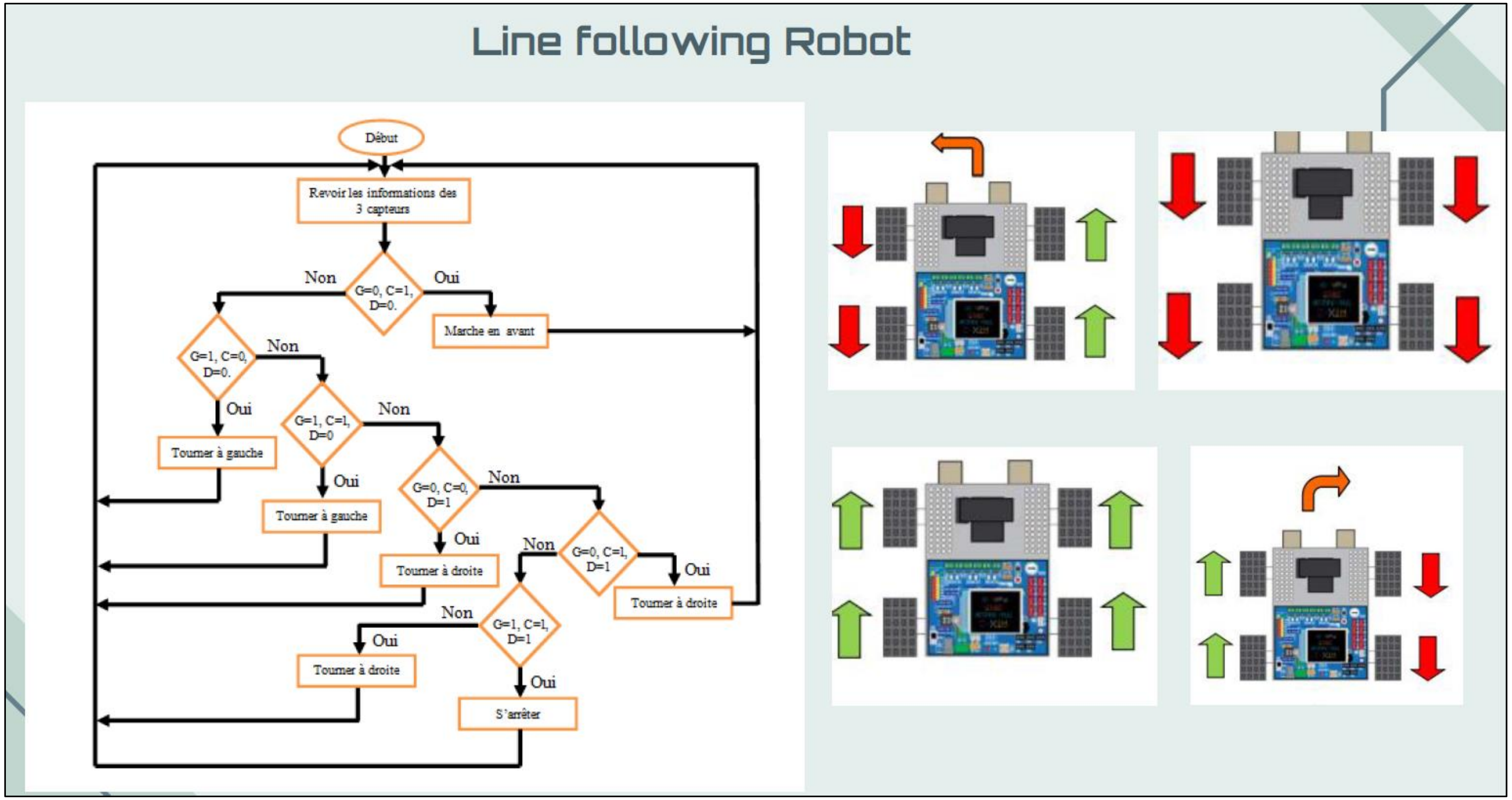
An **Infrared Sensor (IR Sensor)** is a device that detects infrared radiation in its surroundings, commonly used to measure distances, detect objects, or identify heat sources. IR sensors typically emit infrared light and detect its reflection to determine proximity, making them useful in applications like obstacle detection, line following in robots, and motion sensing.

Ultrasonic Sensor

An **Ultrasonic Sensor** is an electronic device that measures distance by emitting high-frequency sound waves (typically around 40 kHz) and calculating the time it takes for the echo to return after bouncing off an object. By using the speed of sound, the sensor can accurately determine the distance to the object based on the time delay. Ultrasonic sensors are widely used in applications like obstacle detection, distance measurement, and autonomous navigation in robotics and vehicles.

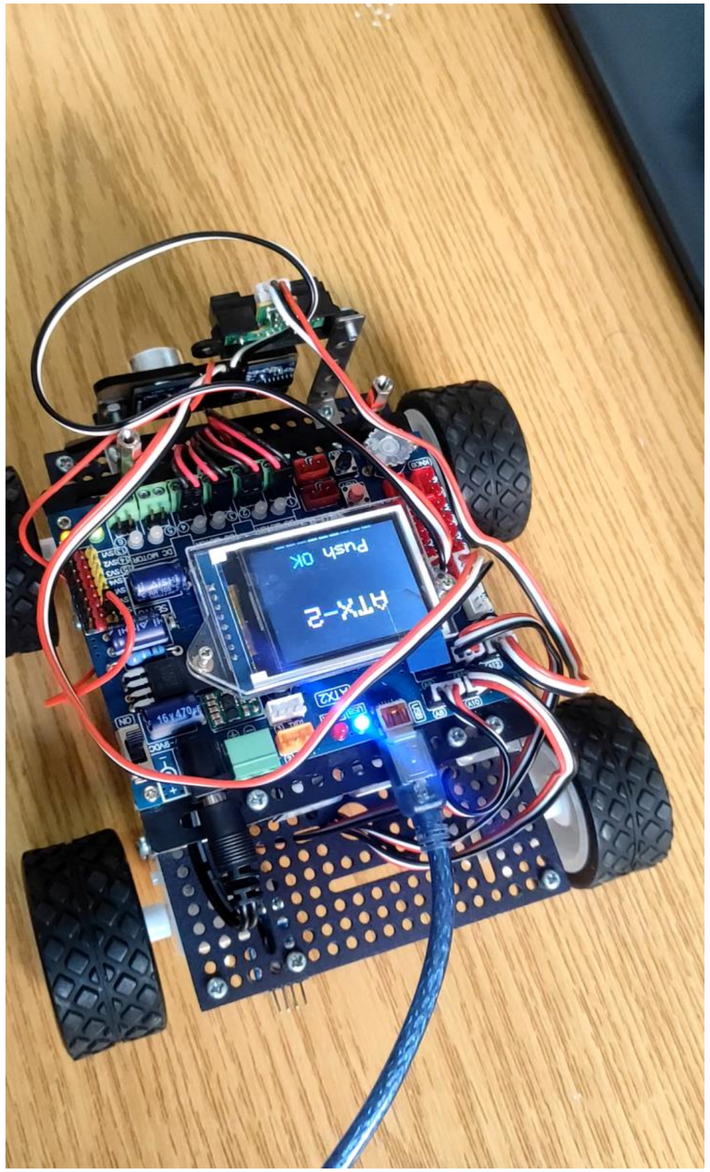


Flowchart

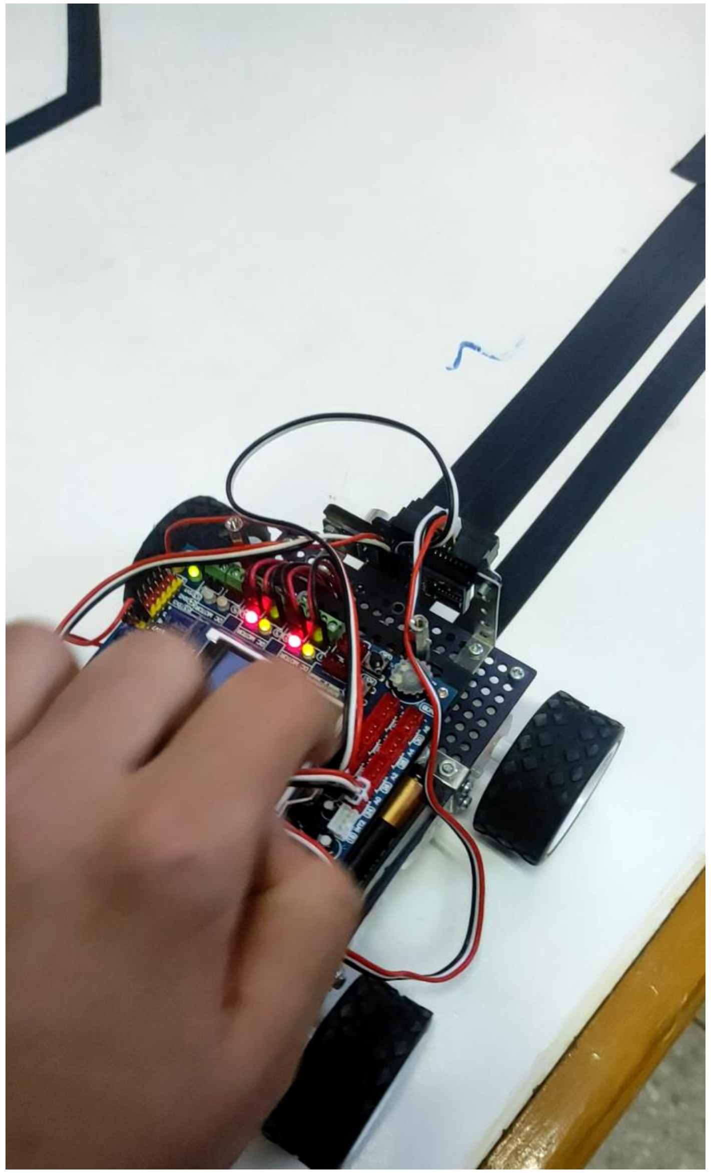


Note : Please refer to my GitHub for the codes.

Demo



obstacle avoidance



Line-following robot

Videos Links:

Line-following robot

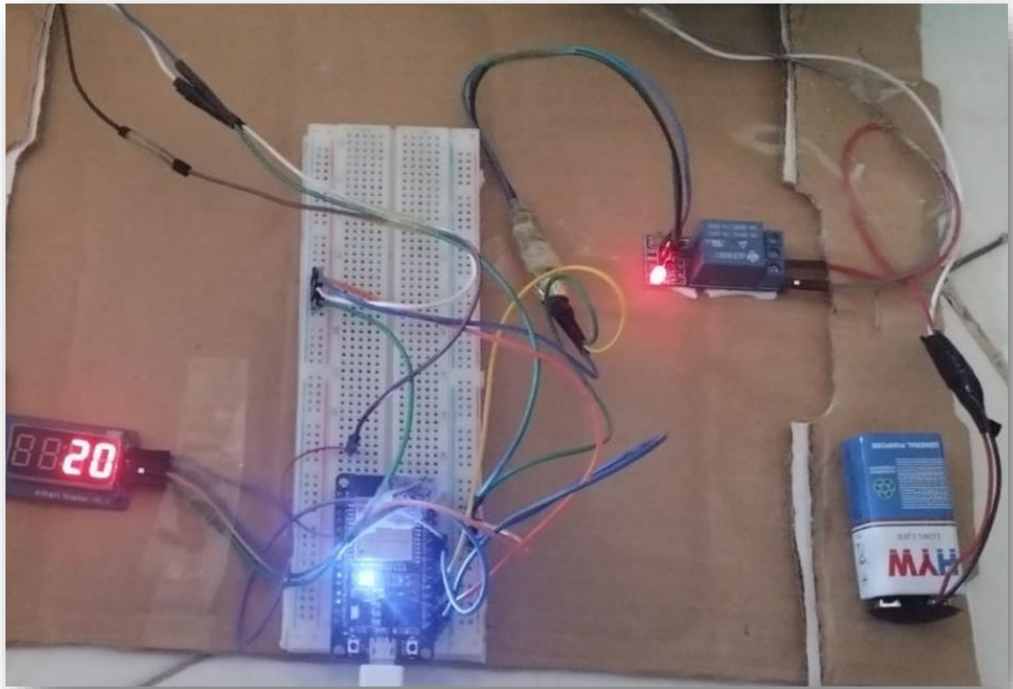
https://drive.google.com/file/d/1KV0WWZoOcOBWxOmh0KbxZZo6hAJDTBM3/view?usp=drive_link

obstacle avoidance

https://drive.google.com/file/d/1JOOLA76BmxDRukCM6F4GrNVRUXKbYjdV/view?usp=drive_link

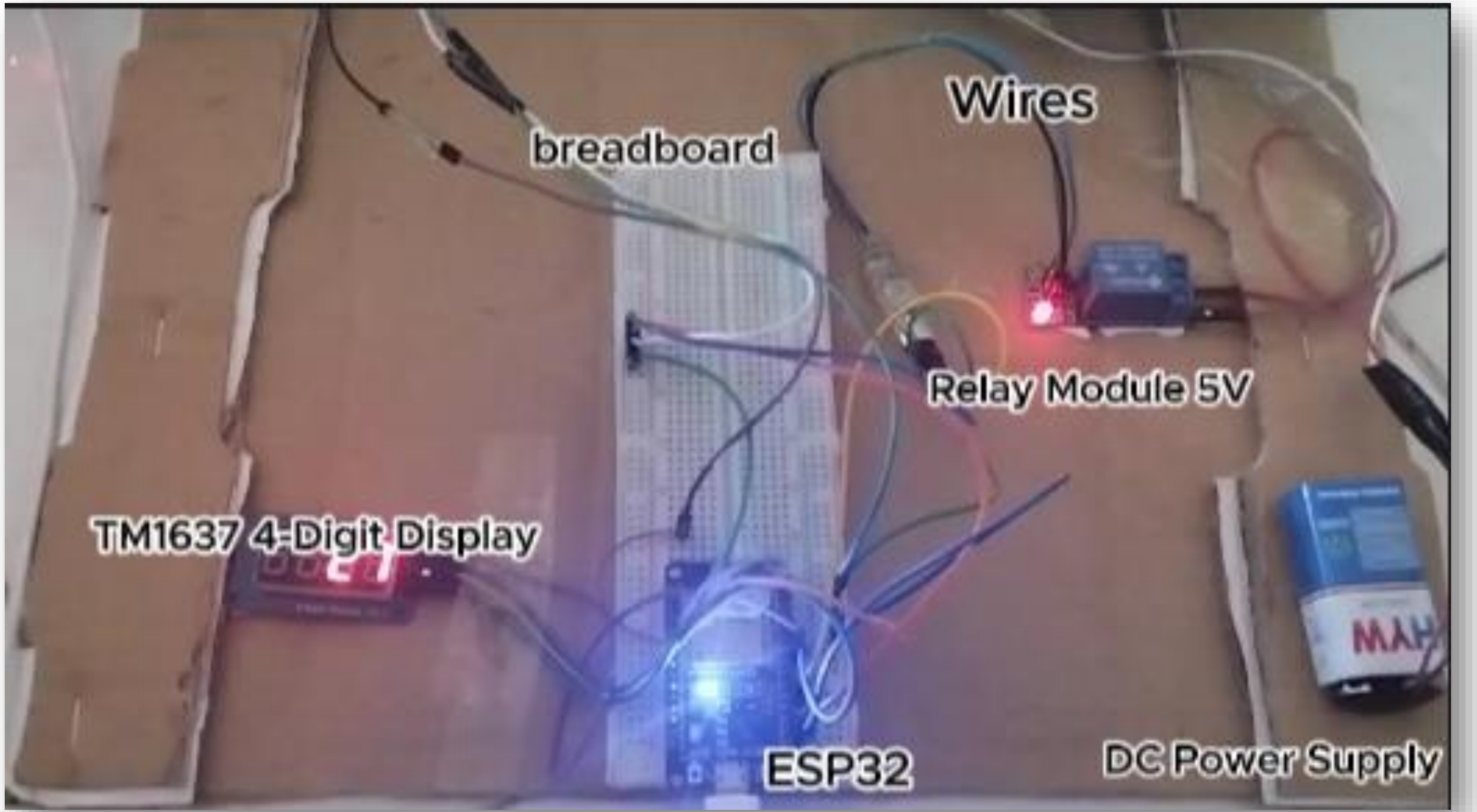
Plant watering System

This application was developed with my younger brother for his high school project, showcasing remote plant watering and care through the Internet of Things (IoT).



Note : Please refer to my GitHub for the codes.

Demo



Video Link: https://drive.google.com/file/d/1Gdzzv7nyzbtiTE3ntwVksM-qPySq1c2Y/view?usp=drive_link

Wiper System Project

This application is a reverse learning project developed after completing Master **Wleed Elshemey's** Udemy course. The course was invaluable for learning the fundamentals of Model-Based Design, covering essential blocks like switches, merge blocks, subsystem masks, function call subsystems, trigger subsystems, and C code generation.

The project models an automotive wiper system to simulate its real-world behavior across various modes: off, low, high, and automatic.

Wiper System Control Inputs:

Inputs	Data Type	Description
WiprMod	Unit8	Wiper mode of operation 0:off 1:Automatic 2:Lowspeed 3:High speed
RainSnsrErr	Boolean	Rain sensor error 0:normal,no error 1:error
WiprspdReq	uint8	Required speed level in case of automatic mode 0 1 2 3 4 5 6 7

Wiper System Control Outputs:

Outputs	Data Type	Description
WiprMotPwmDutyCyc	single	PWM command to the wiper motor
WiprActv	boolean	Indicates if the wiper motor is runnig\activated 0:Stop 1:Running

Wiper System Control Processing:

- **WiperMod:** Wiper mode of operation

0:off

- **WiprMotPwmDutyCyc** =0%

1:Automatic:

In case that **RainSnsErr** is true, then the PWM shall be zero
Else

- WiprSpdReq:Required speed level in case of automatic mode:[0 1 2 3 4 5 6 7]
- Rain sensor to PWM table :[0 40 % 45 % 50 % 55 % 60 % 665 % 70 %]
- It is required to have a smooth PWM command in automatic mode

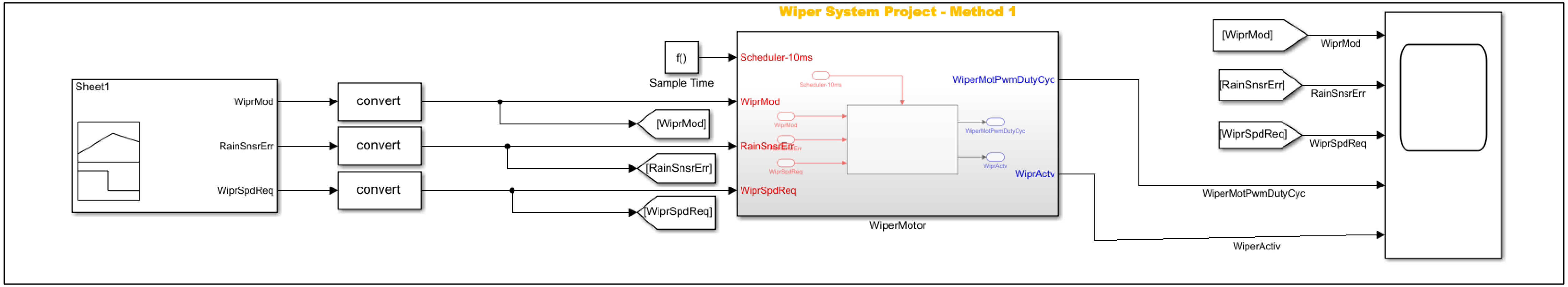
2:Low Speed:

- **WiprMotPwmDutyCyc** =40%

3:High Speed

- **WiprMotPwmDutyCyc** =70%

Wiper System Block



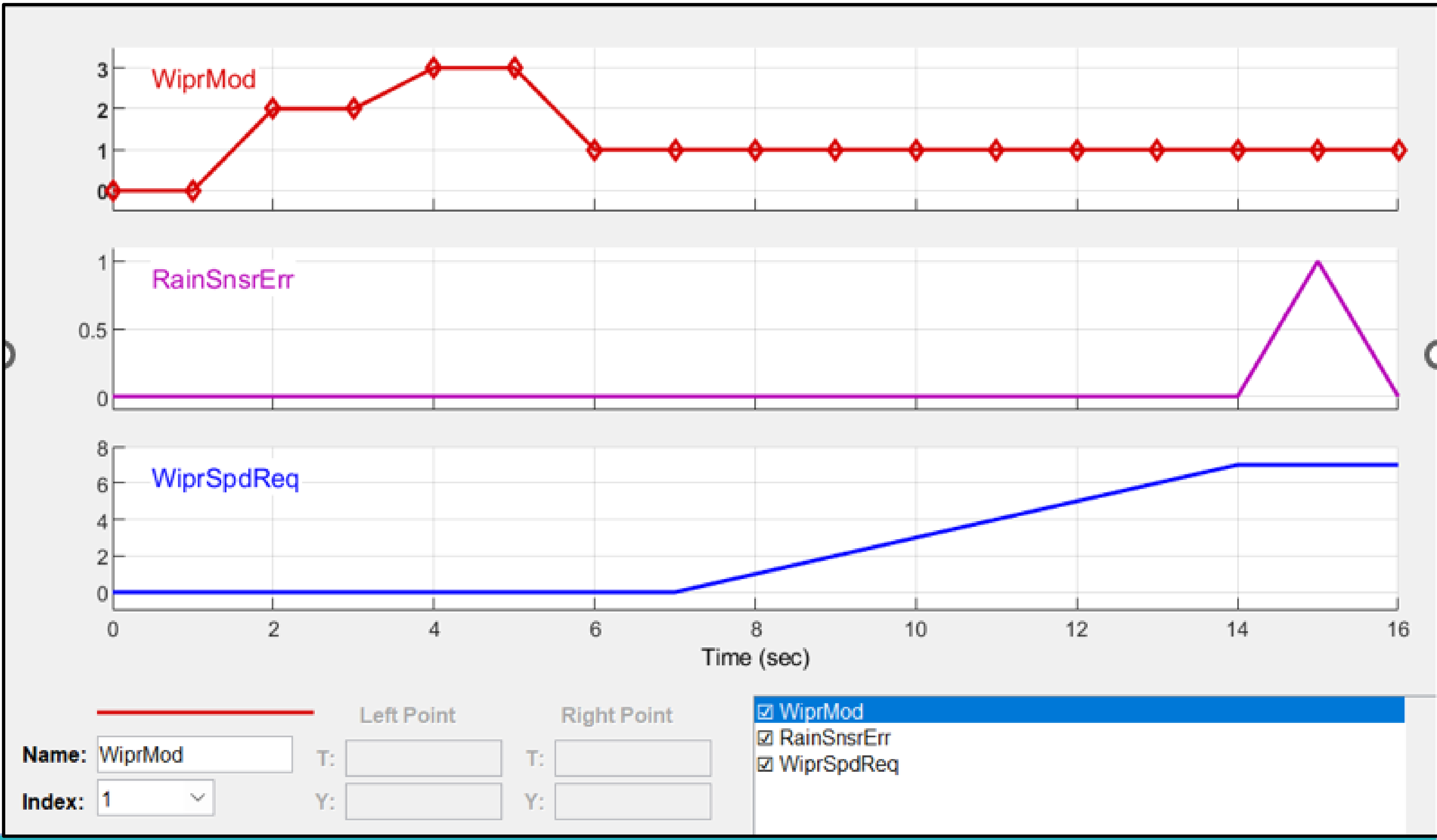
Explanation of the Wiper Block System

Sheet 1 represents a signal builder that contains various values for the three inputs: **wiprMod**, **RainSnsErr**, and **WiprSpdReq**. These inputs are used to run the system and verify if it responds according to our specified requirements.

- ***Excel sheet file***

We created an Excel sheet containing the data, which we then imported into the Signal Builder.

Feuille de calcul Microsoft Excel



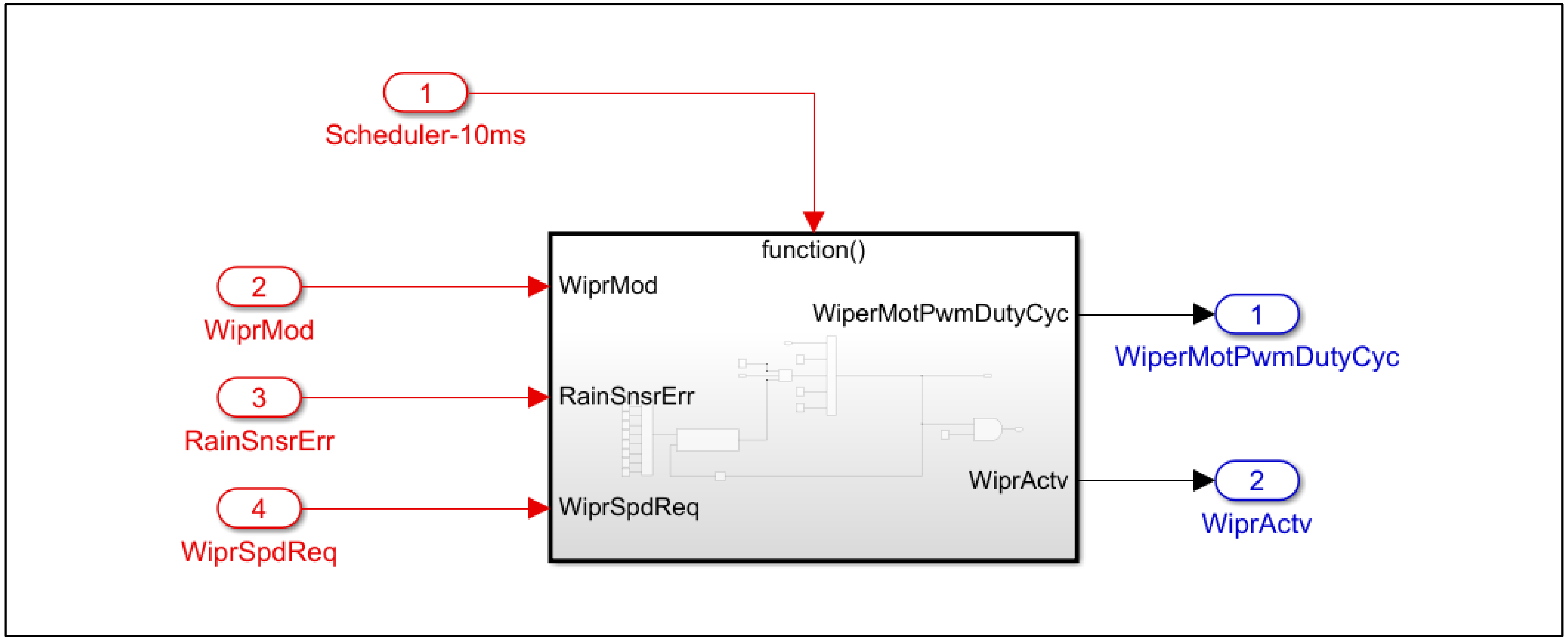
Name: WiprMod

Index: 1

☑ WiprMod
☑ RainSnsrErr
☑ WiprSpdReq

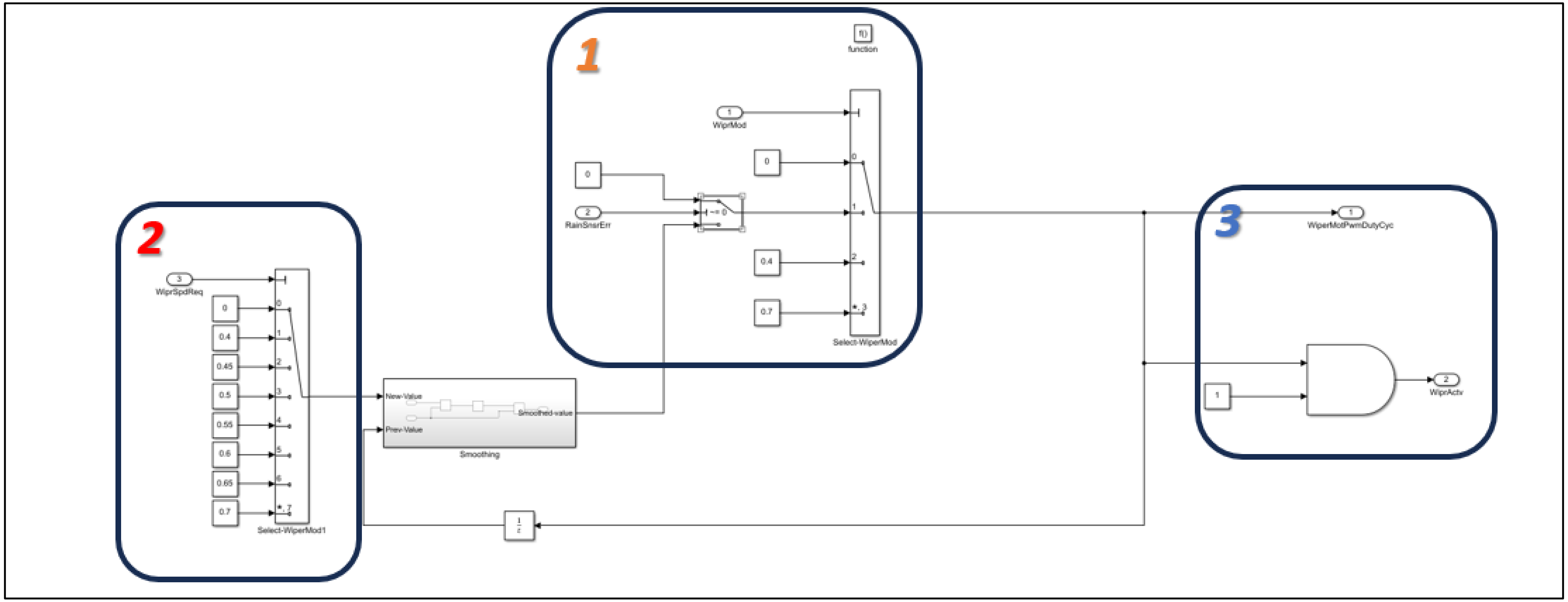
Explanation of the Wiper Block System

- Next, Data Type Converter blocks were added to change signal data types, ensuring compatibility with the WiperMotor.
- Goto/From indicators were used to simplify wiring and improve visibility.
- The Function Call Generator (Sample Time) is used to execute the system every 10 ms.
- The WiperMotor block is a subsystem that includes a function call subsystem within it.



Explanation of the Wiper Block System

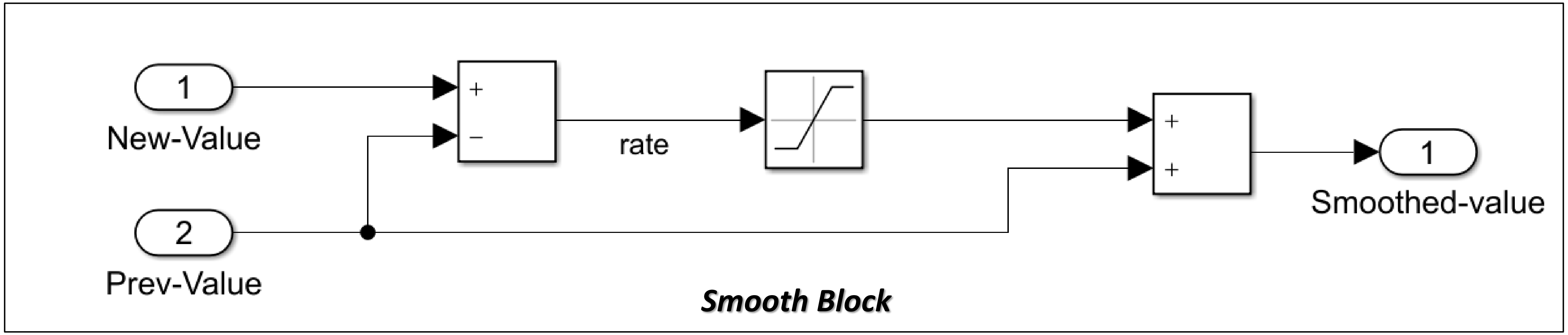
- The function call subsystem translates the requirements for wiper system control into a simulation by processing inputs and outputs using switches, multiport switches, AND logic port, and unit delays.



Explanation of the Wiper Block System

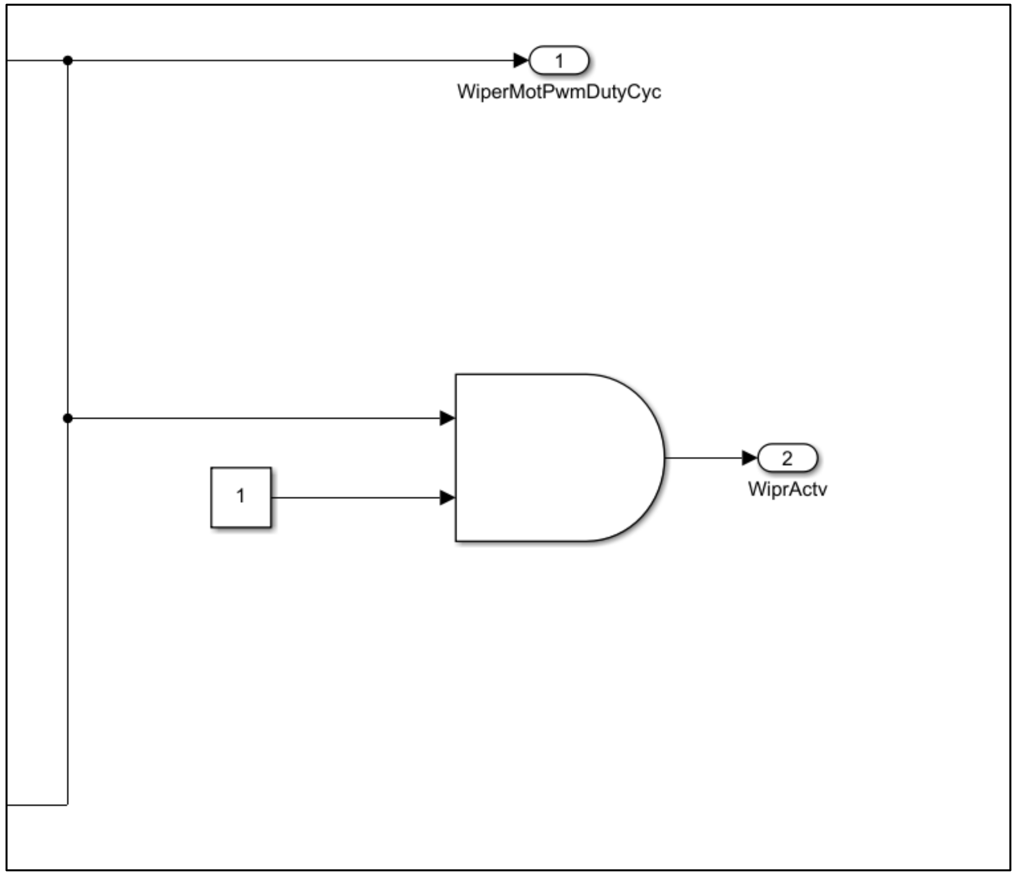
1-The first part corresponds to the wiper modes in a multiport switch. For example, if the switch selects input 2 (0.4), it indicates that the wiper is in low-speed mode, resulting in a duty cycle of 40% for the wiper motor.

2-If the switch selects input 1, it means the system is in automatic mode. In this case, there are seven speed levels corresponding to the following duty cycles: [0, 40%, 45%, 50%, 55%, 60%, 65%, 70%]. This generates a PWM signal, which is then processed through a smooth block interface to compare the new and previous values, followed by saturation to provide a smoothed output.

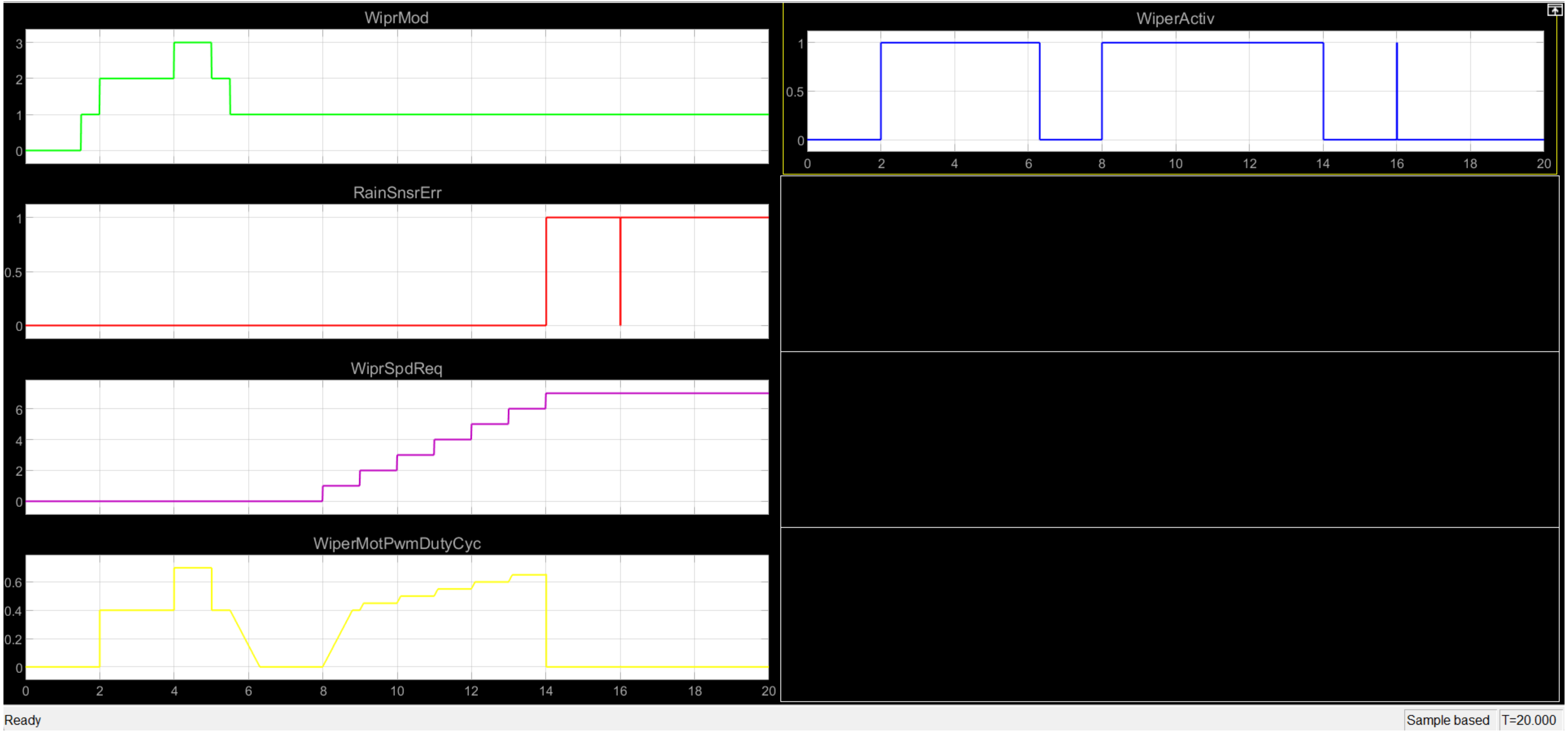


Explanation of the Wiper Block System

3-The WiperMotorPwmDutyCyc output provides the duty cycle based on the wiper mode, while the wiper activation signal indicates whether the wiper motor is running or activated through the AND logic port.



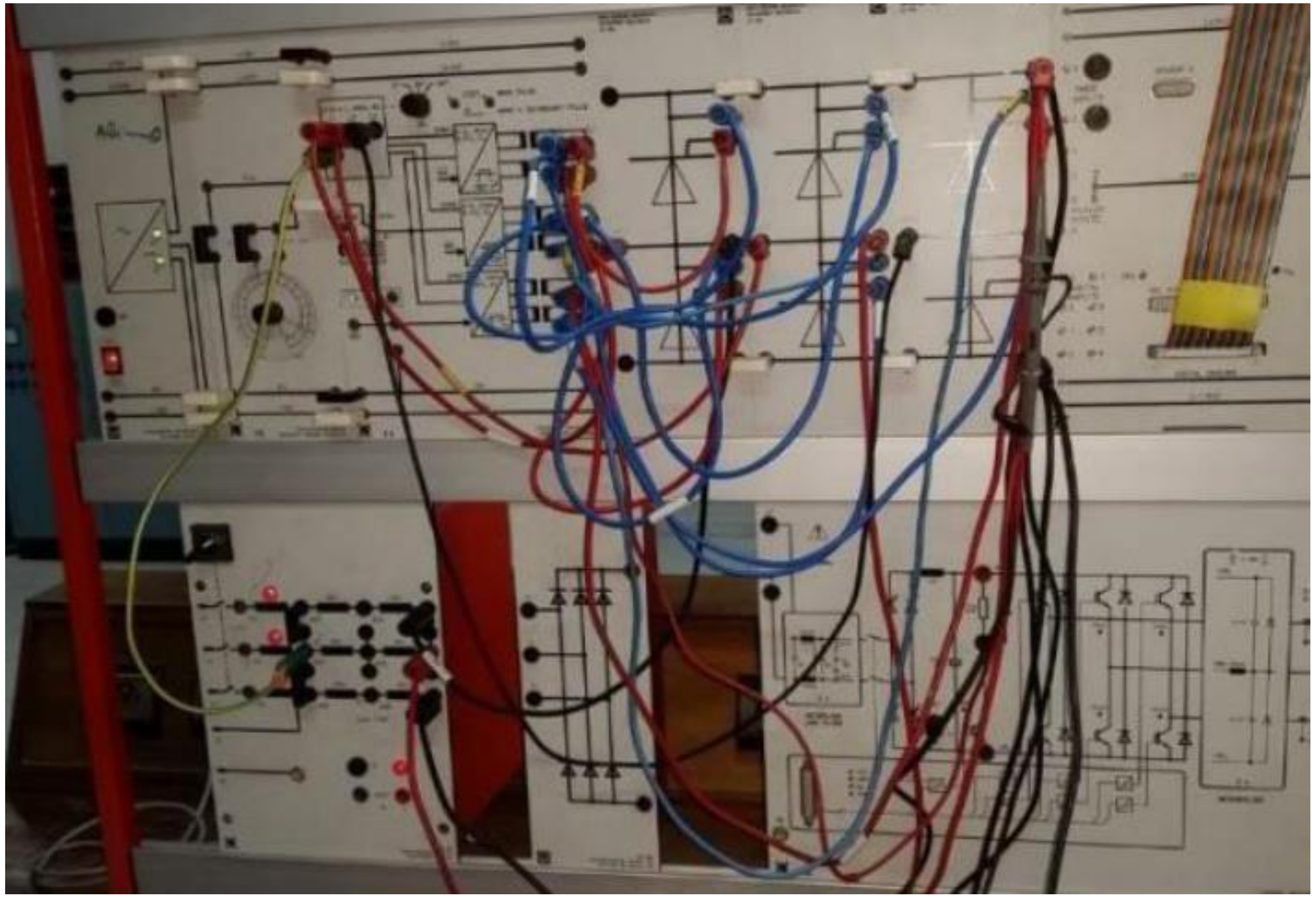
Signals


WiprMod
RainSnsrErr
WiprSpdReq
WiperMotPwmDutyCyc
WiperActiv
Ready
Sample based
T=20.000

Introduction:

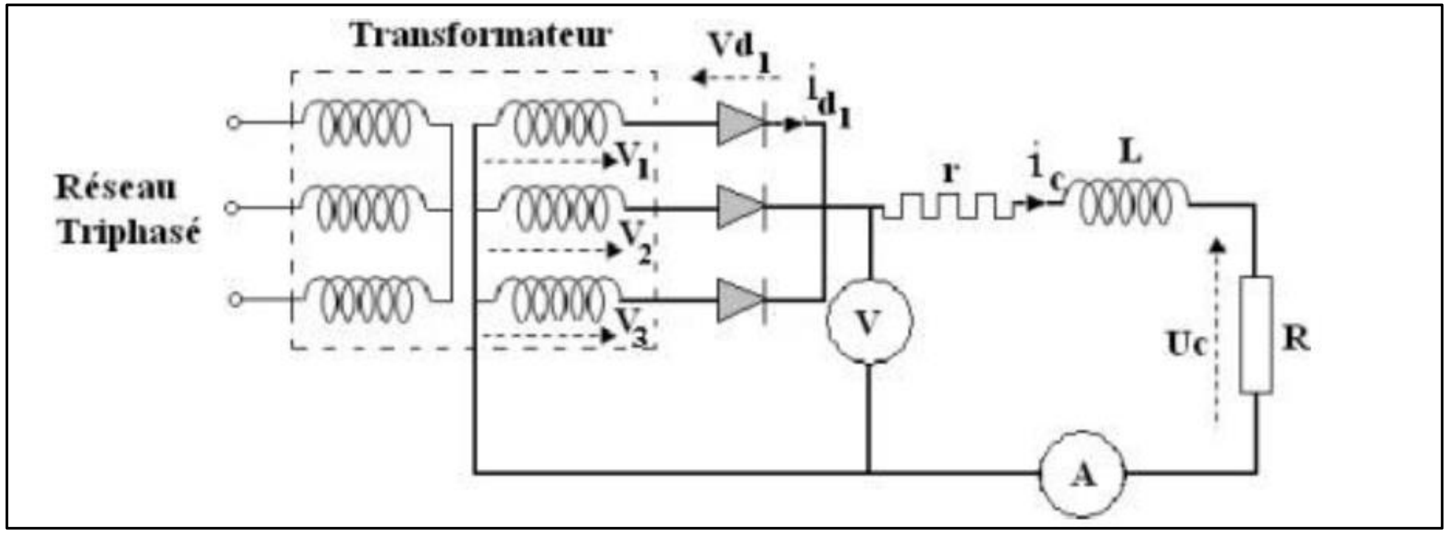
- This project aims to implement controlled and uncontrolled three-phase rectification and visualize the resulting output currents and voltages.

- The role of **uncontrolled and controlled three-phase rectification** is to convert alternating current (AC) into direct current (DC) in electrical systems, but each type offers different levels of control over the output.



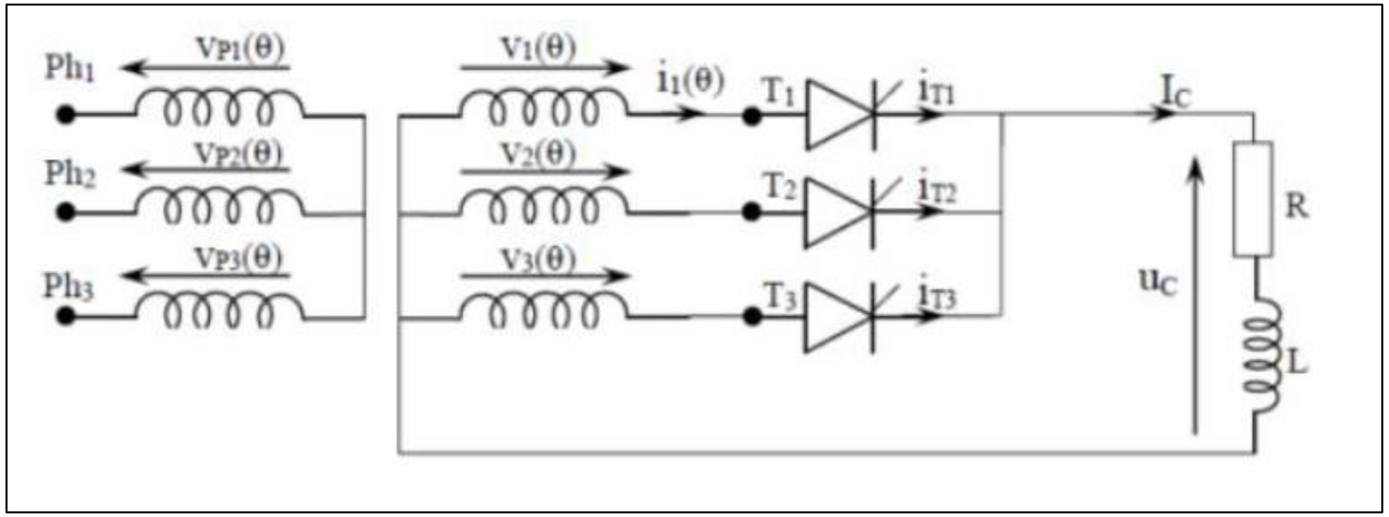
Uncontrolled Three-Phase Rectification

- **Components**: Uses diodes.
- **Control**: No control over output; produces a fixed DC output voltage.
- **Applications**: Ideal for applications requiring a stable, constant DC voltage (e.g., battery charging, basic DC power supplies).
- **Operation**: Simple, cost-effective



Controlled Three-Phase Rectification

- ❑ **Components**: Uses thyristors.
- ❑ **Control**: Allows regulation of the DC output by adjusting the firing angle of the thyristors, which controls when current flows.
- ❑ **Applications**: Suitable for applications needing adjustable DC output.
- ❑ **Operation**: More complex and requires a control circuit.



Link to the file : : https://drive.google.com/file/d/1pZdKvLuZPjDlu7BbJEZcz9fJ3xk5r9Np/view?usp=sharing

Thank you for you Attention

I'm here to collaborate—there's so much more to explore together!