**C-code Generator For RST controller Documentation**

Table of Contents

[**1.** **Abstract** 2](#_Toc48111813)

[**2.** **Problem statement** 2](#_Toc48111814)

[**3.** **Solution design** 3](#_Toc48111815)

[Gui Design 4](#_Toc48111816)

[Calculation engine design 5](#_Toc48111817)

[File Generator 5](#_Toc48111818)

[Simulink model Generator 5](#_Toc48111819)

[C-code Generator 5](#_Toc48111820)

[Ide code generator(for testing) 6](#_Toc48111821)

[**4.** **Validation Method** 7](#_Toc48111822)

[**5.** **Results (validation, Testing)** 8](#_Toc48111823)

[**6.** **Conclusion** 19](#_Toc48111824)

[**7.** **Video link on YouTube** 19](#_Toc48111825)

[**8.** **Appendix** 19](#_Toc48111826)

[**9.** **Contribution** 27](#_Toc48111827)

[**10.** **One Drive Link** 27](#_Toc48111828)

[**11.** **Team members** 27](#_Toc48111829)

# **Abstract**

Objective : developing a configurable and buildable C-code library for generic RST digital controller. Simulation of closed loop control should be done on a simulator or evaluation board (e.g. TivaC EVB) for Arm Cortex M4 platform. Project ends at real time simulation or/and emulation of process and controller in closed loop. We reached the requirements using MATLAB (Gui ,Simulink ,scripts ,code generation) ,keil, code blocks ,excel. We acquired the definition files for the system ,and generated c code for the controller , validated and tested using a lecture example.

# **Problem statement**

The problem is when we need to control a plant with a specific regulation performance and characteristics , we have to calculate the RST parameters using matrices to solve Bezout equation, which requires several calculations and it is an inefficient use of time.

# **Solution design**

In this phase, we take the inputs form the user using the Gui , use them to calculate the RST parameter using the calculation engine , generate the definitions files(confirmation .h, configuration .c ,RST Parameters .h , RST Parameters .c) using file Generator ,then we build a Simulink model with our Design to validate our calculations ,and check if that model is the required one ,then we generate the C-code for the model using the C-code generator after configuring the parameters of The Code generator ,we take the file generated on the ide to build and test the signals to compare with our design on Simulink to validate and test the generated C-code before uploading it on our hardware (the c -code generated from MATLAB is totally hardware independent but we selected a specific hardware to fit our data type with the generated code).

A close up of text on a white surface

Description automatically generated

## Gui Design

Using The MATLAB script that developed by ENG. Ayman ,which take the configuration as input and calculate the RST parameters and visualize the control signal and output signal ,we Developed much more Features as we would show in this Report.

**A screenshot of a cell phone on a table

Description automatically generated**

* **Inputs**
* Plant Configuration
* **Continues**: Ts, Delay ,plant num ,pant denominator.
* **Discreate** : B(z) and A(z).
* Regulation Performance .
* Load Disturbance Frequency.
* Desired Tracking (Omega And Zeta).
* **Outputs**
  + **Button generate definition files generate :**
    - configuration files (input) (configuration .h , configuration .c ).
    - RST Parameters files (out of the calculation engine )( RST Parameters .h , RST Parameters .c).
  + **Generate Automatic C code Button :**
    - Configure and Generate automatically the controller as a C-code file.
    - Generate Simulink model for the whole system to validate the requirement to the system .
    - Generate Simulink model for the controller to make sure the block has the features needed in the controller (filters, buffers).
  + **Generate Simulation C code Button :**
    - Generate separate file for validation and testing the signals (U(k) ,E(k) ,Y(k)) ,that code to simulate the closed loop RST parameters on the system.

## Calculation engine design

By applying Bezout equation on the Regulation and the plant parameters we get the Required R, S and T parameter.

.

## File Generator

In this stage , we generate four files ,two header files and two source files main parameters that we would use them later to generate ,build ,validate and test the code generated.

## Simulink model Generator

We generate two Simulink models:

* System as whole (inputs ,filter ,controller ,plant, Outputs).
* RST Closed loop System only (REQUIRED MODULE).

## C-code Generator

We generate two C-codes:

* first well structured generated C code from the Automatic code generation in MATLAB.
* Second one for testing and validation and can be uploaded on the hardware and works very well as we used it for testing.

## Ide code generator(for testing)

This stage we use the generated a C-code form the previous stage to simulate the closed loop system, that code generate an excel file with the value of Y(k) and U(K) (output and control signals)and plot them with time (k\*Ts) to compare the code generated result with Simulink result.

* We initialization the Gui using one of the Lecture Example as we set the values in the opening function.

**Lecture example:**

* Plant data :d=0 ,B(z)=.1 + .2 ,A (z)= 1 -1.3742 +48875 .
* Tracking Dynamics: Bm(z)=.1 + .2 ,Am(z)= 1 -1.3742 +48875 , Ts=1 s,

.

* Regulation Dynamics P(z)=1 -1.3741 +.4867 ,Ts=1 s, .
* A screenshot of a cell phone

  Description automatically generatedAfter using bezout equation R(z)= 3 -3.94 +1.3141 ,S(z)= 1 - .3742 -.6258 .

A screenshot of a cell phone

Description automatically generated

# **Validation Method**

* First we enter a lecture example into the Gui ,then click generate Definition files and then check configuration .h, configuration .c ,RST Parameters .h , RST Parameters .c ,the header files should countian the definition of the vectors and the source should contain the values of the vectors.
* Then we click generate C-code ,it should generate automatic C-code specific for Cortex M4 and Generic C-code for any system (using the last header and source file generated in the previous step).
  + To validate and test the automatic code generated you can use:
    - SIL(software in the Loop) : as it convert the block into a C code or assembly code and simulate it on the software.
    - S function :it is a Block can convert the C files to MATLAB script and simulate it on the Simulink easily.
  + To validate the Generic code that I wrote to simulate the system:
    - First run the code by any C ide then it would generate an Excel file with the values of Y(K) ,U(k) and time (k\*Ts).
    - Then we can select the columns and plot them on graphs and compare the result with the Gui result.

# **Results (validation, Testing)**

We test this example that I set in the Gui as a default.

A screenshot of a cell phone

Description automatically generated

1- click Generate Definition files the four files

A screenshot of a social media post

Description automatically generated

Figure :configuration.c file

A screenshot of a social media post

Description automatically generated

Figure : configuration .h file

A screenshot of a social media post

Description automatically generated

Figure :RST .c file

A screenshot of a social media post

Description automatically generated

Figure : RST.h file

2-click Generate automatic C code files

The Automatic Code Generation would be generated for arm cortex-M4.

A screenshot of a computer

Description automatically generated

This code can be tested using 3 method

* Open ert\_main.c and controller .c files on any ide for simualtion Add the input step to and take the outputs U,Y values to an excel file and visualize them .

**\*** Code generated **for** Simulink model 'System\_20200611\_1515\_16\_controller'**.**

**\***

**\*** Model version **:** 1.0

**\*** Simulink Coder version **:** 8.14 **(**R2018a**)** 06**-**Feb**-**2018

**\*** C**/**C**++** source code generated on **:** Thu Jun 11 15**:**15**:**19 2020

**\***

**\*** Target selection**:** ert**.**tlc

**\*** Embedded hardware selection**:** NXP**->**Cortex**-**M4

**\*** Code generation objectives**:** Unspecified

**\*** Validation result**:** Not run

**\*/**

#include **<**stddef**.**h**>**

#include **<**stdio**.**h**>** **/\*** This ert\_main**.**c example uses printf**/**fflush **\*/**

#include "System\_20200611\_1515\_16\_controller.h" **/\*** Model**'**s header file **\*/**

#include "rtwtypes.h"

**/\***

**\*** Associating rt\_OneStep with a real**-**time clock or interrupt service routine

**\*** is what makes the generated code "real-time"**.** The **function** rt\_OneStep is

**\*** always associated with the base rate of the model**.** Subrates are managed

**\*** by the base rate from inside the generated code**.** Enabling**/**disabling

**\*** interrupts and floating point context switches are target specific**.** This

**\*** example code indicates where these should take place relative to executing

**\*** the generated code step **function.** Overrun behavior should be tailored to

**\*** your application needs**.** This example simply sets an error status in the

**\*** real**-**time model and returns from rt\_OneStep**.**

**\*/**

void rt\_OneStep**(**int Ts**,**FILE **\***fp**,**int k**);**

void rt\_OneStep**(**int Ts**,**FILE **\***fp**,**int k**)**

**{**

static boolean\_T OverrunFlag **=** false**;**

**/\*** Disable interrupts here **\*/**

**/\*** Check **for** overrun **\*/**

**if** **(**OverrunFlag**)** **{**

rtmSetErrorStatus**(**System\_20200611\_1515\_16\_cont\_M**,** "Overrun"**);**

**return;**

**}**

OverrunFlag **=** true**;**

**/\*** Save FPU context here **(if** necessary**)** **\*/**

**/\*** Re**-**enable timer or interrupt here **\*/**

**/\*** Set model inputs here **\*/**

System\_20200611\_1515\_16\_contr\_U**.**into**=**1**;**

**if(**k**<**100**){**

System\_20200611\_1515\_16\_contr\_U**.**into2**=**0**;**

**}**

**else{**

System\_20200611\_1515\_16\_contr\_U**.**into2**=-**.25**;**

**}**

**/\*** Step the model **\*/**

System\_20200611\_1515\_16\_controller\_step**();**

**/\*** Get model outputs here **\*/**

**if(**k%1==0){

printf**(**"%d ,%f ,%f\n"**,**k**,**System\_20200611\_1515\_16\_contr\_Y**.**Outform2 **,**System\_20200611\_1515\_16\_contr\_Y**.**Outform**);**

fprintf**(**fp**,** "%d,%d,%f,%f\n"**,** k**,** k**\***Ts**,**System\_20200611\_1515\_16\_contr\_Y**.**Outform2**,**System\_20200611\_1515\_16\_contr\_Y**.**Outform**);**

**}**

**/\*** Indicate task complete **\*/**

OverrunFlag **=** false**;**

**/\*** Disable interrupts here **\*/**

**/\*** Restore FPU context here **(if** necessary**)** **\*/**

**/\*** Enable interrupts here **\*/**

**}**

**/\***

**\*** The example "main" **function** illustrates what is required by your

**\*** application code to initialize**,** execute**,** and terminate the generated code**.**

**\*** Attaching rt\_OneStep to a real**-**time clock is target specific**.** This example

**\*** illustrates how you do this relative to initializing the model**.**

**\*/**

int\_T main**(**int\_T argc**,** const char **\***argv**[])**

**{**

**/\*** Unused arguments **\*/**

**(**void**)(**argc**);**

**(**void**)(**argv**);**

**/\*** Initialize model **\*/**

System\_20200611\_1515\_16\_controller\_initialize**();**

**/\*** Attach rt\_OneStep to a timer or interrupt service routine with

**\*** period 1.0 seconds **(**the model**'**s base sample time**)** here**.** The

**\*** call syntax **for** rt\_OneStep is

**\***

**\*** rt\_OneStep**();**

**\*/**

fflush**((**NULL**));**

FILE **\*** fp **=** fopen**(**"mydata.csv"**,** "w"**);**

**if(**fp **==** NULL**){**

printf**(**"Couldnt open file"**);**

**return;**

**}**

int Ts**=**1**;**

**for** **(**int i **=**0**;**i**<**300**;**i**=**i**+**1 **)** **{**

rt\_OneStep**(**Ts**,** fp **,**i**);**

**}**

**/\*** Disable rt\_OneStep**()** here **\*/**

**/\*** Terminate model **\*/**

System\_20200611\_1515\_16\_controller\_terminate**();**

fclose**(**fp**);**

**return** 0**;**

**}**

**/\***

**\*** File trailer **for** generated code**.**

**\***

**\*** **[**EOF**]**

**\*/**

**/\***

**\*** The example "main" **function** illustrates what is required by your

**\*** application code to initialize**,** execute**,** and terminate the generated code**.**

**\*** Attaching rt\_OneStep to a real**-**time clock is target specific**.** This example

**\*** illustrates how you do this relative to initializing the model**.**

**\*/**

int\_T main**(**int\_T argc**,** const char **\***argv**[])**

**{**

**/\*** Unused arguments **\*/**

**(**void**)(**argc**);**

**(**void**)(**argv**);**

**/\*** Initialize model **\*/**

System\_20200611\_1515\_16\_controller\_initialize**();**

**/\*** Attach rt\_OneStep to a timer or interrupt service routine with

**\*** period 1.0 seconds **(**the model**'**s base sample time**)** here**.** The

**\*** call syntax **for** rt\_OneStep is

**\***

**\*** rt\_OneStep**();**

**\*/**

fflush**((**NULL**));**

FILE **\*** fp **=** fopen**(**"mydata.csv"**,** "w"**);**

**if(**fp **==** NULL**){**

printf**(**"Couldnt open file"**);**

**return;**

**}**

int Ts**=**1**;**

**for** **(**int i **=**0**;**i**<**300**;**i**=**i**+**1 **)** **{**

rt\_OneStep**(**Ts**,** fp **,**i**);**

**}**

**/\*** Disable rt\_OneStep**()** here **\*/**

**/\*** Terminate model **\*/**

System\_20200611\_1515\_16\_controller\_terminate**();**

fclose**(**fp**);**

**return** 0**;**

**}**

**/\***

**\*** File trailer **for** generated code**.**

**\***

**\*** **[**EOF**]**

**\*/**

The Result:

A screenshot of a computer

Description automatically generated

* Then generate the Sil module and build it in the module A screenshot of a cell phone

  Description automatically generated

The Result:

A close up of text on a white background

Description automatically generated

Figure : SIL module output

A close up of text on a white background

Description automatically generated

Figure : design output

A close up of text on a white background

Description automatically generated

Figure :compare between them

3- click generate Simulation C files

This file can run on code block or any ide to simulate the closed loop control system as this code generate an Exeal sheet with the control signal and the output signals.

A screenshot of a computer

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a cell phone

Description automatically generated

# **Conclusion**

we were facing a problem which was modeling a controller for a specific plant was a complex problem then it was discovered the designing a controller using was possible but it still required lots of calculation and included a high percentage of error so it was better to let computers do the calculation ,so we design a computer program which take the plant Transfer function and the required performance parameters to generate the required transfer functions as well as the corresponding code which is specific for Arm Cortex-M4 with few adjustments.

# **Video link on YouTube**

<https://www.youtube.com/watch?v=9USYxJDnOng&feature=youtu.be>

# **Appendix**

**GitHub link:**

<https://github.com/girgismicheal/Control-Project>

%last edit 10/6/2020 by girgis micheal

%at the beginning we initialize the model with the lecture example as

%following

%at the beginning we initialize the model with the lecture example as following

% --- Executes just before RST\_PP\_v2 is made visible.

function RST\_PP\_v2\_OpeningFcn(hObject, eventdata, handles, varargin)

% Choose default command line output for RST\_PP\_v2

handles.output = hObject;

% Update handles structure

guidata(hObject, handles);

set(handles.edit1, 'string','1');%Ts

%continuous plant

set(handles.edit2, 'string','0');%delay %3 0

set(handles.edit3, 'string','1');%num

set(handles.edit4, 'string','[10 1]');%den

set(handles.checkbox1, 'value',0);%discrete plant

%discrete plant

set(handles.edit5, 'string','[0 0.1 0.2]');%Bp

set(handles.edit6, 'string','[1 -1.3 0.42]');%Ap

set(handles.checkbox2, 'value',0);%continuous design criteria

%regulation P(z)

set(handles.edit7, 'string','0.4');%w0

set(handles.edit8, 'string','0.9');%zeta

set(handles.edit11, 'string','[1 -1.3741 0.4867]');%P (just in case)

%tracking Bm(z)/Am(z)

set(handles.edit9, 'string','0.5');%w0

set(handles.edit10, 'string','0.9');%zeta

set(handles.edit12, 'string','[0.0927 0.0687]');%Bm

set(handles.edit13, 'string','[1 -1.2451 0.4066]');%Am (just in case)

set(handles.edit14, 'string','1');%Hr

set(handles.edit15, 'string','1');%Hs

set(handles.edit16, 'string','0');%disturbance cosine frequency

%set(handles.edit16, 'string','[1 -1]');%disturbance den

set(handles.checkbox3, 'value',0);

%set(handles.checkbox3, 'string','Unused');%cancel zeros

pushbutton2\_Callback(0, 0, handles);

Gui initialization

% --- Executes on button press in pushbutton5.

function pushbutton5\_Callback(hObject, eventdata, handles) %Generate Definition Files

% hObject handle to pushbutton5 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

[Ts Bp Ap Hr Hs dist P Bm Am]=acquire\_data(handles);

%create file configration.h

fileP=fopen('configration.h','w');

fprintf(fileP,'#ifndef \_CONFIG\_H \n#define \_CONFIG\_H\_\n');

fclose(fileP);

fileP=fopen('configration.h','a');

fprintf(fileP,'#define delay= %f ;', str2num(get(handles.edit2,'string')));

fprintf(fileP,'\n#define Ts=%f ;', Ts);

fprintf(fileP,'\nextern const float B[%d], A[%d], P[%d],Am[%d],Bm[%d];',numel(Bp),numel(Ap),numel(Ap),numel(Am),numel(Bm));

fprintf(fileP,'\n#endif');

fclose(fileP);

%create configration.c

fileP=fopen('configration.c','w');

fprintf(fileP, '#include \"configration.h\"\n');

fprintf(fileP, 'const float P[%d]=',numel(P));

fprintf(fileP, '{');

for k=1:numel(P)-1

fprintf(fileP, [num2str(P(k)) ',']);

end

fprintf(fileP, [num2str(P(k+1)) '};\n']);

%B(z)

fileP=fopen('configration.c','a');

fprintf(fileP, 'const float B[%d]=',numel(Bp));

fprintf(fileP, '{');

for k=1:numel(Bp)-1

fprintf(fileP, [num2str(Bp(k)) ',']);

end

fprintf(fileP, [num2str(Bp(k+1)) '};\n']);

%A(z)

fileP=fopen('configration.c','a');

fprintf(fileP, 'const float A[%d]=',numel(Ap));

fprintf(fileP, '{');

for k=1:numel(Ap)-1

fprintf(fileP, [num2str(Ap(k)) ',']);

end

fprintf(fileP, [num2str(Ap(k+1)) '};\n']);

fclose(fileP);

%Am(z)

fileP=fopen('configration.c','a');

fprintf(fileP, 'const float Am[%d]=',numel(Am));

fprintf(fileP, '{');

for k=1:numel(Am)-1

fprintf(fileP, [num2str(Am(k)) ',']);

end

fprintf(fileP, [num2str(Am(k+1)) '};\n']);

fclose(fileP);

%Bm(z)

fileP=fopen('configration.c','a');

fprintf(fileP, 'const float Bm[%d]=',numel(Bm));

fprintf(fileP, '{');

for k=1:numel(Bm)-1

fprintf(fileP, [num2str(Bm(k)) ',']);

end

fprintf(fileP, [num2str(Bm(k+1)) '};\n']);

fclose(fileP);

Bps=[]; Bpu=[];

if get(handles.checkbox3, 'value') %cancel stable zeros

[Bps Bpu]=separate\_B(Bp);

else %don't cancel

Bps=1; Bpu=Bp;

end

%SOLUTION

%Ap Hs S + Bp Hr R = P (no zeros cancelled)

S\_has\_integrator=has\_no\_integrator(Ap);

Hs\_full=[];

if S\_has\_integrator

Hs\_full=conv(Hs, [1 -1]);

else

Hs\_full=Hs;

end

A=conv(Ap, Hs\_full)';

B=conv(Bpu, Hr)';

nA=size(A); nA=nA(1)-1;%vertical vector size, power of z = size-1

nB=size(B); nB=nB(1)-1;

A=padarray(A, nB-1, 0, 'post');

B=padarray(B, nA-1, 0, 'post');

M=[];

for k=1:nB

M=[M A];

A=circshift(A, 1);

end

for k=(nB+1):(nB+nA)

M=[M B];

B=circshift(B, 1);

end

%M

n=size(M); n=n(1);%M is a square matrix

nP=size(P); nP=nP(2);

P\_full=padarray(P', n-nP, 0, 'post');

SR=M\P\_full;

%Extract R, S, T

R=SR((nB+1):(nB+nA))';

S=SR(1:nB)';

if S\_has\_integrator

S=conv(S, [1, -1]);

end

S=conv(S, Bps);

T=P/sum(Bpu);%(no zeros cancelled)

set(handles.edit17, 'string',['[' num2str(R) ']']);%full R S T values for Update plant & Simulink buttons

set(handles.edit18, 'string',['[' num2str(S) ']']);

set(handles.edit19, 'string',['[' num2str(T) ']']);

R=conv(R, Hr);

S=conv(S, Hs);

%create file RST.h

fileP=fopen('RST\_parameters.h','w');

fprintf(fileP,'#ifndef \_RST\_H \n#define \_RST\_H\_\n');

fclose(fileP);

fileP=fopen('RST\_parameters.h','a');

fprintf(fileP,'extern const float R[%d], S[%d], T[%d];',numel(R),numel(S),numel(T));

fprintf(fileP,'\n#endif');

fclose(fileP);

%create RST.c

fileP=fopen('RST\_parameters.c','w');

fprintf(fileP, '#include \"RST\_parameters.h\"\n');

%R(z)

fprintf(fileP, 'const float R[%d]=',numel(R));

fprintf(fileP, '{');

for k=1:numel(R)-1

fprintf(fileP, [num2str(R(k)) ',']);

end

fprintf(fileP, [num2str(R(k+1)) '};\n']);

%S(z)

fileP=fopen('RST\_parameters.c','a');

fprintf(fileP, 'const float S[%d]=',numel(S));

fprintf(fileP, '{');

for k=1:numel(S)-1

fprintf(fileP, [num2str(S(k)) ',']);

end

fprintf(fileP, [num2str(S(k+1)) '};\n']);

%T(z)

fileP=fopen('RST\_parameters.c','a');

fprintf(fileP, 'const float T[%d]=',numel(T));

fprintf(fileP, '{');

for k=1:numel(T)-1

fprintf(fileP, [num2str(T(k)) ',']);

end

fprintf(fileP, [num2str(T(k+1)) '};\n']);

fclose(fileP);

Definition files Generator code

function pushbutton6\_Callback(hObject, eventdata, handles)%Generate Simulation C Files

[Ts Bp Ap Hr Hs dist P Bm Am]=acquire\_data(handles);

Bps=[]; Bpu=[];

if get(handles.checkbox3, 'value') %cancel stable zeros

[Bps Bpu]=separate\_B(Bp);

else %don't cancel

Bps=1; Bpu=Bp;

end

%SOLUTION

%Ap Hs S + Bp Hr R = P (no zeros cancelled)

S\_has\_integrator=has\_no\_integrator(Ap);

Hs\_full=[];

if S\_has\_integrator

Hs\_full=conv(Hs, [1 -1]);

else

Hs\_full=Hs;

end

A=conv(Ap, Hs\_full)';

B=conv(Bpu, Hr)';

nA=size(A); nA=nA(1)-1;%vertical vector size, power of z = size-1

nB=size(B); nB=nB(1)-1;

A=padarray(A, nB-1, 0, 'post');

B=padarray(B, nA-1, 0, 'post');

M=[];

for k=1:nB

M=[M A];

A=circshift(A, 1);

end

for k=(nB+1):(nB+nA)

M=[M B];

B=circshift(B, 1);

end

%M

n=size(M); n=n(1);%M is a square matrix

nP=size(P); nP=nP(2);

P\_full=padarray(P', n-nP, 0, 'post');

SR=M\P\_full;

%Extract R, S, T

R=SR((nB+1):(nB+nA))';

S=SR(1:nB)';

if S\_has\_integrator

S=conv(S, [1, -1]);

end

S=conv(S, Bps);

T=P/sum(Bpu);%(no zeros cancelled)

set(handles.edit17, 'string',['[' num2str(R) ']']);%full R S T values for Update plant & Simulink buttons

set(handles.edit18, 'string',['[' num2str(S) ']']);

set(handles.edit19, 'string',['[' num2str(T) ']']);

R=conv(R, Hr);

S=conv(S, Hs);

%TS and samples

fileP=fopen('main.c','w');

fprintf(fileP,['#include <stdio.h>\n#include <stdlib.h>\nint main()\n{\nFILE \* fp = fopen("mydata.csv", "w");\nif(fp == NULL){\nprintf("Couldnt open file");\nreturn;\n}\n']);

fprintf(fileP,['float Ts=' num2str(Ts) ';\n']);

fprintf(fileP,['int n\_samples=' num2str(1000) ';\n']);

fprintf(fileP,['float Ref[' num2str(numel(T)) ']' ';\n']);

fprintf(fileP,['float Uk[' num2str(numel(S)) ']' ';\n']);

fprintf(fileP,['float Ek[' num2str(numel(Bp)) ']' ';\n']);

fprintf(fileP,['float yk[' num2str(max(numel(conv(S,A)),numel(R))) ']' ';\n']);

Simulation file Generator code

%to calculate y(k)

fileP=fopen('main.c','a');

NUm=conv(A,S)

BE=Bp/NUm(1)

NUm=NUm/-NUm(1)

fprintf(fileP, 'yk[0]=');

for k=1:numel(BE)

if(BE(k)>0)fprintf(fileP, ['+' num2str(BE(k)) '\*Ek[' num2str(k-1) ']']);

else

fprintf(fileP, [num2str(BE(k)) '\*Ek[' num2str(k-1) ']']);

end

end

for k=1:numel(NUm)-1

if(NUm(k+1)>0)fprintf(fileP, ['+' num2str(NUm(k+1)) '\*yk[' num2str(k) ']' ] );

else fprintf(fileP, [num2str(NUm(k+1)) '\*yk[' num2str(k) ']' ] );

end

end

fprintf(fileP,';\n');

fprintf(fileP,['\nif(k==1)yk[0]=.02\*yk[0];\nif(k==100)yk[0]=yk[0]+.28;\n']);

fprintf(fileP,['printf(%cy[%c%c] = %c%c%c%c%c, k, yk[0]);\nfprintf(fp, %c%c%c,%c%c,%c%c,%c%c%c%c%c, k, k\*Ts,-Uk[0],-yk[0]);\n'],'"','%','d','%','f','\n','"','"','%','d','%','f','%','f','%','f','\n','"');

fclose(fileP);

%shift old values

fileP=fopen('main.c','a');

fprintf(fileP,['for(int i=0;i<' num2str(numel(T)-1) ';i++)Ref[i+1]=Ref[i];\n']);

fprintf(fileP,['for(int i=0;i<' num2str(numel(S)-1) ';i++)Uk[i+1]=Uk[i];\n']);

fprintf(fileP,['for(int i=0;i<' num2str(numel(Bp)-1) ';i++)Ek[i+1]=Ek[i];\n']);

fprintf(fileP,['for(int i=0;i<' num2str(max(numel(conv(S,A)),numel(R))-1) ';i++)yk[i+1]=yk[i];\n}']);

fprintf(fileP,['fclose(fp);\nreturn 0;\n}']);

fclose(fileP);

%initailize the arrays

fileP=fopen('main.c','a');

fprintf(fileP,['for(int i=1;i<' num2str(numel(T)) ';i++)Ref[i]=0;\n']);

fprintf(fileP,['for(int i=1;i<' num2str(numel(S)) ';i++)Uk[i]=0;\n']);

fprintf(fileP,['for(int i=1;i<' num2str(numel(Bp)) ';i++)Ek[i]=0;\n']);

fprintf(fileP,['for(int i=1;i<' num2str(max(numel(conv(S,A)),numel(R))) ';i++)yk[i]=0 ;\n']);

fclose(fileP);

%simulation loop

fileP=fopen('main.c','a');

fprintf(fileP,['for(int k=0;k<n\_samples;++k)\n{\n//simulate\n']);

fprintf(fileP,['Ref[0]=1;\n']);

%to calc e(k)

fileP=fopen('main.c','a');

fprintf(fileP, 'Ek[0]=');

for k=1:numel(T)

if(T(k)>0)fprintf(fileP, ['+' num2str(T(k)) '\*Ref[' num2str(k-1) ']']);

else

fprintf(fileP, [num2str(T(k)) '\*Ref[' num2str(k-1) ']']);

end

end

for k=1:numel(R)

if(R(k)>0)fprintf(fileP, ['+' num2str(R(k)) '\*yk[' num2str(k-1) ']' ] );

else fprintf(fileP, [num2str(R(k)) '\*yk[' num2str(k-1) ']' ] );

end

end

fprintf(fileP,';\n');

fclose(fileP);

%to calculate U(k)

fileP=fopen('main.c','a');

NUm=S/-S(1)

fprintf(fileP, 'Uk[0]=Ek[0]');

for k=1:numel(NUm)-1

if(NUm(k+1)>0)fprintf(fileP, ['+' num2str(NUm(k+1)) '\*yk[' num2str(k) ']' ] );

else fprintf(fileP, [num2str(NUm(k+1)) '\*yk[' num2str(k) ']' ] );

end

end

fprintf(fileP,';\n');

fclose(fileP);

Automatic code Generator code

% --- Executes on button press in pushbutton4.

function pushbutton4\_Callback(hObject, eventdata, handles)%Generate Automatic Code Files

% hObject handle to pushbutton4 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% hObject handle to pushbutton3 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

[Ts Bp Ap Hr Hs dist P Bm Am]=acquire\_data(handles);

R=str2num(get(handles.edit17, 'string'));%full R S T values

S=str2num(get(handles.edit18, 'string'));

T=str2num(get(handles.edit19, 'string'));

%global it\_counter;

%it\_counter=it\_counter+1;

c=clock;

name=sprintf('System\_%d%02d%02d\_%02d%02d\_%02d', c(1), c(2), c(3), c(4), c(5), round(c(6)));

nameofcont=sprintf('System\_%d%02d%02d\_%02d%02d\_%02d\_controller', c(1), c(2), c(3), c(4), c(5), round(c(6)));%controller

sim\_time=100; t\_ref=5; t\_dist=50;

str\_Ts=num2str(Ts);

x=0; y=0;

sys=new\_system(name);

sys2=new\_system(nameofcont);%controller

set\_param(name, 'stoptime',num2str(sim\_time));

set\_param(nameofcont, 'stoptime',num2str(sim\_time));%controller

x=x+0; y=y+100;%step position

add\_block('simulink/Sources/Step', [name '/Reference'], 'position',[x, y, x+30, y+30],...

'time',num2str(t\_ref\*Ts), 'sampletime',str\_Ts);

x=x+30;%step width

x=x+35;%spacing

add\_block('simulink/Discrete/Discrete Filter', [nameofcont '/Bm(z)//Am(z)'], 'position',[x, y, x+195, y+30],...

'numerator',['[' num2str(Bm) ']'], 'denominator',['[' num2str(Am) ']'], 'sampletime',str\_Ts);

x=x+195;%tracking width

x=x+35;%spacing

add\_block('simulink/Discrete/Discrete Filter', [nameofcont '/T(z)'], 'position',[x, y, x+195, y+30],...

'numerator',['[' num2str(T) ']'], 'denominator','[1]', 'sampletime',str\_Ts);

x=x+195;%T width

add\_line(nameofcont, 'Bm(z)//Am(z)/1', 'T(z)/1');

x=x+25;%spacing

add\_block('simulink/Math Operations/Sum', [nameofcont '/Sum1'], 'position',[x, y+5, x+20, y+5+20],...

'inputs','|+-');

x=x+20;%sum width

add\_line(nameofcont, 'T(z)/1', 'Sum1/1');

x=x+20;%spacing

add\_block('simulink/Discrete/Discrete Filter', [nameofcont '/1//Hs(z)'], 'position',[x, y, x+195, y+30],...

'numerator','[1]', 'denominator',['[' num2str(Hs) ']'], 'sampletime',str\_Ts);

x=x+195;%Hs width

add\_line(nameofcont, 'Sum1/1', '1//Hs(z)/1');

x=x+35;%spacing

add\_block('simulink/Discrete/Discrete Filter', [nameofcont '/1//S(z)'], 'position',[x, y, x+195, y+30],...

'numerator','[1]', 'denominator',['[' num2str(S) ']'], 'sampletime',str\_Ts);

x=x+195;%S width

add\_line(nameofcont, '1//Hs(z)/1', '1//S(z)/1');

%add\_line(name, 'Sum1/1', '1//S(z)/1');

x=x+330;%make space for possible continuous plant

add\_block('simulink/Math Operations/Sum', [nameofcont '/Sum2'], 'position',[x+100+30, y+5, x+100+50, y+5+20],...

'inputs','++|');

x=x+20;%sum width

x=x-285;%return till after S

if get(handles.checkbox1, 'value')%continuous plant

add\_block('simulink/Discrete/Zero-Order Hold', [name '/Zero-Order Hold'], 'position',[x-800, y+30, x+35-800, y+60]);

x=x+35;%ZOH width

%add\_line(name, '1//S(z)/1', 'Zero-Order Hold/1');

x=x+35;%spacing

add\_block('simulink/Continuous/Transport Delay', [name '/Plant Delay'], 'position',[x-800, y+30, x+30-800, y+60],...

'delaytime',get(handles.edit2, 'string'));

x=x+30;%delay width

add\_line(name, 'Zero-Order Hold/1', 'Plant Delay/1');

x=x+35;%spacing

add\_block('simulink/Continuous/Transfer Fcn', [name '/Gp(s)'], 'position',[x-800, y+30, x+130-800, y+80],...

'numerator',get(handles.edit3,'string'), 'denominator',get(handles.edit4,'string'));

x=x+120;%Gp width

add\_line(name, 'Plant Delay/1', 'Gp(s)/1');

add\_line(name, 'Gp(s)/1', 'Sum2/2');

else%discrete plant

add\_block('simulink/Discrete/Discrete Filter', [nameofcont '/Bp(z)//Ap(z)'], 'position',[x+100, y+5, x+195+100, y+35],...

'numerator',['[' num2str(Bp) ']'], 'denominator',['[' num2str(Ap) ']'], 'sampletime',str\_Ts);

x=x+195;%plant width

%add\_line(name, '1//S(z)/1', 'Bp(z)//Ap(z)/1');

add\_line(nameofcont, 'Bp(z)//Ap(z)/1', 'Sum2/2');

end

x=x+35;%spacing till sum2

x=x-135;%new branch

add\_block('simulink/Sources/Sine Wave', [name '/Disturbance'], 'position',[x-1200, 170, x-1200+30, 200],...

'amplitude','-0.25', 'frequency',get(handles.edit16,'string'), 'phase','pi/2', 'sampletime',str\_Ts);

x=x+30;%disturbance width

x=x+40;%spacing

add\_block('simulink/Continuous/Transport Delay', [name '/Disturbance Delay'], 'position',[x-1200, 170, x-1200+30, 200],...

'delaytime',num2str(t\_dist\*Ts));

x=x+30;%delay width

add\_line(name, 'Disturbance/1', 'Disturbance Delay/1');

x=x+125;%spacing from dist. delay till mux

add\_block('simulink/Signal Routing/Mux', [name '/Mux'], 'position',[x-800, y, x-800+5, y+38]);

x=x+5;%mux width

x=x+35;%spacing

add\_block('simulink/Sinks/Scope', [name '/Scope'], 'position',[x-800, y-5, x-800+30, y+42]);

%add\_line(name, '1//S(z)/1', 'Mux/1');

add\_line(name, 'Mux/1', 'Scope/1');

%add\_line(name, 'Sum2/1', 'Scope/1');

%feedback blocks

x=x-465; y=y+110;%from scope left edge to R

block\_R=add\_block('simulink/Discrete/Discrete Filter', [nameofcont '/R(z)'], 'position',[x, y, x+195, y+30],...

'numerator',['[' num2str(R) ']'], 'denominator','[1]', 'sampletime',str\_Ts, 'orientation','left');

%all=get(get\_param(block\_R, 'handle'))

x=x-195;%R width

%add\_line(name, 'Sum2/1', 'R(z)/1');

x=x-35;%spacing

add\_block('simulink/Discrete/Discrete Filter', [nameofcont '/Hr(z)'], 'position',[x, y, x+195, y+30],...

'numerator',['[' num2str(Hr) ']'], 'denominator','[1]', 'sampletime',str\_Ts, 'orientation','left');

x=x-195;

add\_line(nameofcont, 'R(z)/1', 'Hr(z)/1');

add\_line(nameofcont, 'Hr(z)/1)', 'Sum1/2');

%add\_line(name, 'R(z)/1)', 'Sum1/2');

add\_block('simulink/Sinks/Out1', [nameofcont '/Outform'],'position',[1600, 100, 1620, 130])

add\_block('simulink/Sinks/Out1', [nameofcont '/Outform2'],'position',[1600, 50, 1620, 80])

add\_block('simulink/Sources/In1', [nameofcont '/into'],'position',[0, 100, 20, 130])

add\_block('simulink/Sources/In1', [nameofcont '/into2'],'position',[1200, y-300, 1220, y-300+30])

add\_line(nameofcont, '1//S(z)/1', 'Bp(z)//Ap(z)/1');

add\_line(nameofcont, 'into/1', 'Bm(z)//Am(z)/1');

add\_line(nameofcont, 'into2/1', 'Sum2/1');

add\_line(nameofcont, 'Sum2/1', 'Outform/1');

add\_line(nameofcont, 'Sum2/1', 'R(z)/1');

add\_line(nameofcont, '1//S(z)/1', 'Outform2/1');

add\_block('built-in/Subsystem', [name '/controller'],'position',[100, 85, 200, 200])

Simulink.BlockDiagram.copyContentsToSubsystem...

(nameofcont, [name +'/controller/'])

add\_line(name, 'Disturbance Delay/1', 'controller/2');

add\_line(name, 'Reference/1', 'controller/1','autorouting','smart');

%add\_line(name, 'Sum2/1', 'controller/2','autorouting','smart');

add\_line(name, 'controller/1', 'Mux/1');

if get(handles.checkbox1, 'value')%continuous plant

%add\_line(name, 'controller/1', 'Zero-Order Hold/1','autorouting','smart');

else%discrete plant

%add\_line(name, 'controller/1', 'Bp(z)//Ap(z)/1','autorouting','smart');

end

add\_line(name, 'controller/2', 'Mux/2','autorouting','smart');

load\_system(nameofcont)

cs = getActiveConfigSet(nameofcont);

cs.set\_param('Solver', 'FixedStepauto');

switchTarget(cs,'ert.tlc',[])

set\_param(nameofcont, 'ProdHWDeviceType', 'NXP->Cortex-M4')

rtwbuild(nameofcont)

open\_system(sys);

open\_system(sys2);

lineHandles = find\_system(gcs,'FindAll','On','SearchDepth',1,'Type','Line');

# **Contribution**

Girgis Micheal: Searching and Understanding of the problem ,designing the solution ,Assign the team roles , Finding the Examples(Test cases), Testing and Validation Video , Software Finishing (calculate the modules).

Pierre Nabil: Code Generation Component.

John Bahaa: Testing and Validation.

Hazem Mousa: Definition Files Generation Component ,build on Keil.

Ibrahim Shoukry: Documentation ,Simulation code .

# **One Drive Link**

<https://onedrive.live.com/?authkey=%21AJX5Rx4CKnc5U68&id=80BDBBE808EDCE73%21106&cid=80BDBBE808EDCE73>

# **Team members**

Girgis Micheal

Pierre Nabil

Hazem Mousa

Ibrahim Shoukry

John Bahaa