

Memorandum

To: Bill Bennett (Intuitive Surgical)
From: Juan Nunez
Date: 19 November, 2020

Subject: Delivery of “Fan Control” Coding Challenge.

Summary:

The following is a delivery of the Fan Control software component for you and your team’s assessment as a part of my interview process.

Artifacts:

The following artifacts are provided as a part of this delivery:

<div><div>Name</div><div><div><div>_Docs</div><div><div>FanControl_DesignDocument.pdf</div><div>FanControl_Memorandum.pdf</div><div>IntuitiveCodingChallenge_FanControl.docx</div></div></div><div><div>_Snapshot_FanControlUI_Exe</div><div><div>platforms</div><div><div>FanControlUI.exe</div><div>Qt5Core.dll</div><div>Qt5Gui.dll</div><div>Qt5Widgets.dll</div></div></div></div><div><div>FanControlComponent</div><div>FanControlComponent_Gtest</div><div>FanControlComponent_Lib</div><div>FanControlComponent_UI</div><div>FanControlUI_Exe</div><div>_ReadMe.txt</div><div>Build_All.cmd</div><div>Build_FanControl.cmd</div><div>Build_FanControl_Gtest.cmd</div><div>Build_FanControl_Lib.cmd</div><div>Build_FanControl_UI.cmd</div><div>clean_x64.cmd</div></div></div></div> <td><ul style="list-style-type: none">• _Docs:<ul style="list-style-type: none">○ FanControl_Memorandum.pdf: This document.○ FanControl_DesignDoc.pdf: A simplified design document for the FanControlComponent.○ IntuitiveCodingChallenge_FanControl.docx: Original challenge Instructions.• _Snapshot_FanControlUI_exe: Folder containing a snapshot of a working FanControl Demo and supporting DLLs.• FanControlComponent: Solution for the stand alone FanControl Component.• FanControlComponent_Gtest: Solution for the FanControl with GTest.• FanControlComponent_Lib: Solution for the FanControl Library.• FanControlComponent_UI: Solution for the QT UI.• FanControlUI_Exe: Folder containing the Demo executable and needed QT DLLs.• _ReadMe.txt: Text file pointing the reader to this document.• Build_All.cmd: Does a full clean and rebuild of both Debug and Release for all the solutions; Also deleting x64 folders.• Build_FanControl.cmd: Builds the Stand Alone FanControl Component as an executable.• Build_FanControl_Lib.cmd: Builds the FanControl Library.• Build_FanControl_UI.cmd: Builds the Demo with QT GUI, essentially building the FanControl library and QT UI. The Executable is copied into the FanControlUI_Exe folder.• Build_FanControl_Gtest.cmd: Builds the Fan Control Component with GTest executable.• clean_x64.cmd: Deletes the various x64 folders created and populated during the build process.</td>	<ul style="list-style-type: none">• _Docs:<ul style="list-style-type: none">○ FanControl_Memorandum.pdf: This document.○ FanControl_DesignDoc.pdf: A simplified design document for the FanControlComponent.○ IntuitiveCodingChallenge_FanControl.docx: Original challenge Instructions.• _Snapshot_FanControlUI_exe: Folder containing a snapshot of a working FanControl Demo and supporting DLLs.• FanControlComponent: Solution for the stand alone FanControl Component.• FanControlComponent_Gtest: Solution for the FanControl with GTest.• FanControlComponent_Lib: Solution for the FanControl Library.• FanControlComponent_UI: Solution for the QT UI.• FanControlUI_Exe: Folder containing the Demo executable and needed QT DLLs.• _ReadMe.txt: Text file pointing the reader to this document.• Build_All.cmd: Does a full clean and rebuild of both Debug and Release for all the solutions; Also deleting x64 folders.• Build_FanControl.cmd: Builds the Stand Alone FanControl Component as an executable.• Build_FanControl_Lib.cmd: Builds the FanControl Library.• Build_FanControl_UI.cmd: Builds the Demo with QT GUI, essentially building the FanControl library and QT UI. The Executable is copied into the FanControlUI_Exe folder.• Build_FanControl_Gtest.cmd: Builds the Fan Control Component with GTest executable.• clean_x64.cmd: Deletes the various x64 folders created and populated during the build process.
--	---

Build Environment:

- Visual Studio 19
 - QT Visual Studio 2019 Tools (Extension)
 - Test Adapter for Google Test (Installer)
- gRPC MSVC142_64
 - https://github.com/thommyho/gRPC_windows/releases/tag/v1.22.0
 - Unzip under: ..\FanControl\FanControlComponent\gRpc\
- QT 5.15.1
 - <https://www.qt.io/> (Open Source Qt)

Environment Variables:

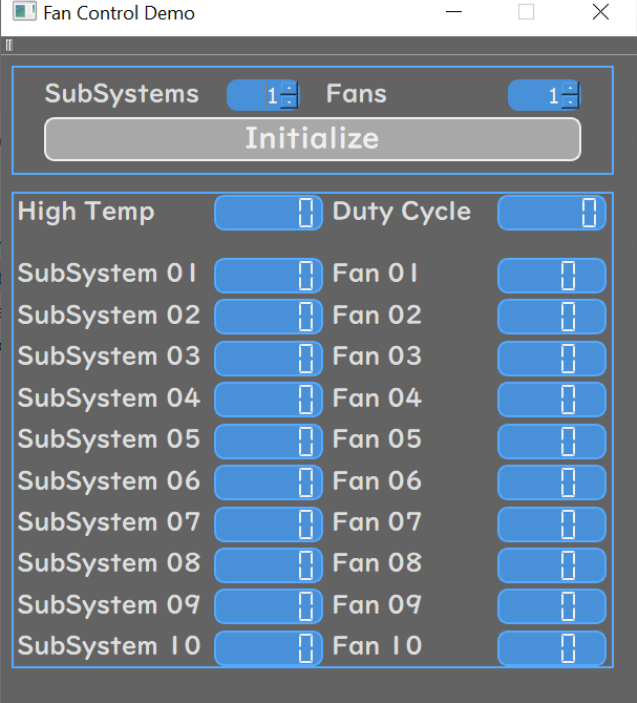
- MSBuild: Ensure the path to MSBuild has been added to the windows PATH environment variable.
 - **Example:** C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\MSBuild\Current\Bin\

Building:

- With the build environment setup, either open a solution via Visual Studio or run the desired [*.cmd] command script.

Fan Control Demo:

The Fan Control Demo exercises the Fan Control Component by instantiating one or more SubSystems. The Fan Control will then repeatedly pull temperatures from each SubSystem. The Fan Control Component will keep track of the highest temperature, calculate the base fan duty cycle for the highest temperature, calculate the specific PWM Count for each fan and finally program the fan with the PWM Count value.



Inputs:

- SubSystems [1:10]: The number of SubSystems to instantiate.
- Fans: [1:10]: The number of Fans available.

Note: The demo randomly assigns the instance number, between [1:10] during instantiation. They are not necessarily sequential.

Button:

- The single button has 3 functions:
 - **Initialize:** Available only once, instantiates the FanControl Component and SubSystem(s).
 - **Start:** Commands the FanControl to start pulling SubSystem temperatures.
 - **Stop:** Stops the FanControl.

Note: Once initialized, the “Inputs” become disabled.

Display:

- High Temp: The current highest temperature.
- Duty Cycle: The base duty cycle base on the High Temp Value.
- SubSystems [1:10]: The current temp for the SubSystem.
- Fan [1:10]: The current PWM Count for the Fan.

Fan Control Stand Alone:

The Fan Control Component solution is setup to build as an executable, as opposed to a library, which contains a main that is hard-coded to instantiate 10 SubSystems and 10 Fans. Once running, the FanControl will print the current high temperature and the base duty cycle.

```
FanControl::ctor() - USING MOCK MEMORY ADDRESSES FOR REGISTERS
FanControl::updateFans(): CurTemp=[25], DC=[20]
SubSystem[1]::RunServer() - Server listening on localhost:50051
SubSystem[2]::RunServer() - Server listening on localhost:50052
SubSystem[3]::RunServer() - Server listening on localhost:50053
SubSystem[4]::RunServer() - Server listening on localhost:50054
SubSystem[5]::RunServer() - Server listening on localhost:50055
SubSystem[6]::RunServer() - Server listening on localhost:50056
SubSystem[7]::RunServer() - Server listening on localhost:50057
SubSystem[8]::RunServer() - Server listening on localhost:50058
SubSystem[9]::RunServer() - Server listening on localhost:50059
SubSystem[10]::RunServer() - Server listening on localhost:50060
-----
P - Menu
G - Go
S - Stop
E - exit
-----
g
FanControl::updateFans(): CurTemp=[30], DC=[28]
FanControl::updateFans(): CurTemp=[30.1], DC=[28.16]
FanControl::updateFans(): CurTemp=[30.2], DC=[28.32]
FanControl::updateFans(): CurTemp=[30.3], DC=[28.48]
FanControl::updateFans(): CurTemp=[s30.4], DC=[28.64]
FanControl::updateFans(): CurTemp=[30.3], DC=[28.48]
-----
P - Menu
G - Go
S - Stop
```

Menu:

- P – Print the menu.
- G – Commands the FanControl to begin pulling temperatures from the SubSystems and updating fans.
- S – Stop FanControl.
- E – Exit the demo.