Struts4Embedded
MP3 Player w/ Remote Access via WIFI
Using STM32F769i Discover Board & ChibiOS
Software Architecture Overview

Board: STM32F769i-Discovery Board
Clock: M=25, N=432, P=2, Q=2, R=2 at 216 MHz and PLLI2S Clock Params: N=429, P=2, Q=2, R=2, PLLI2SR=19,HCLK=216Mhz
RTOS: ChibiOS 19.1.0: Serial Driver/VCP (w/ SD1 for chprintf & SD5 to communicate with WIFI-Module) , objFIFO, Dynamic Pool, & Mailbox
HAL: ChibiOS HAL Layer: ADCD1 (sampled via TGRO on TIM4 10KHz), I2CD4, SDCD2, MMCD2 (MMCD2 w/ SDC), SAI1, UART1, UART5, PAL & PAL Callback
IDE: ChibiStudio
3rd Party Libs: JSMN JSON, Hashmap, Madlib
github repo: https://github.com/abusous2000/MP3PlayerUsingSTM32F7AndChibiOS

ActionEvent is similar to the implementation of the *Command Pattern.* Simply it's a data structure containing data about the event(i.e. which button was clicked, its name, the triggering source) plus a pointer to a function that later will be executed to service this event.
This AEs are created in the ISR (step 3.0), and pumped into a queue (3.2) when triggered by user events. Later the main thread (step 3.3) pulls one AE at a time from the queue, and executes its action.
The **BEST** design if the queue is a blocking priority queue (not yet implemented), and the AEs are pulled out based on a user defined priority, and later executed by Worker Executor Thread Pool that blocks on this priority queue. In the Java domain, this pattern has been implemented in Spring Integration and Camel frameworks; please consult online documentation for more details about this pattern.

WifiModule ESP8266
Runs simple C++ based RESTful web service, server & configured via Arduino IDE

Main Thd
Stack Size: 0x1000

Audio Player Thd
Stack Size: 0x2000

«SDCardDriverITF»

FatFS

«AudioPlayerITF»

MadLib
«MP3 Decoder»

«MMCDriver»

Worker Thread Pool
Stack Size: 1024
Size: 2 threads

Blinker Thd
Stack Size: 1024

«CodecDriverITF»

AEs are pulled & executed in a Worker Thread one at a time

ActionEvent Queue

«Adds ActionEvent per Click/WS To AE Queue»

«ADCDriver»

CPal(GPIO Helper Class)

«ssd1306_Driver»

«SAIDriver»

«I2CDriver»

«SDCDriver»

WiFi Module Thd
Stack Size: 1024

Button ISR

ADC ISR

Sends JSON Payload via UART5

«DMADriver»

Sends RESTful WS invocations via CURL or JS (in a browser) w/ JSON Payloads

«uses ADCD1»

«uses»

«starts»

«starts»

«starts»

«notifies»

«decodes»

«decodes»

«notifies driver of user actions»

«notifies driver of user actions»

«initi., sets volume»

«Displays MP3 Player status info»

«uses I2CD1»

«Mounts SD, File System & Iterates over files»

«init. & configs MMC and SDC»

updates memory with decoded MP3 Audio from SD Card

«uses I2CD4»

«uses SDCD2»

«uses I2CD4 w/ DMA1 on channel 7»

«uses SAID1 w/ DMA2 on channel 1»

«uses SDCD2 w/ DMA2 on channel 0»

1.0   1.1   1.11   1.12   1.13
1.2   1.21   1.22   1.23   1.24
2.0   2.1   2.2   2.3   2.4
3.0   3.1   3.2   3.3   3.4

Sends RESTful WS invocations via CURL or JS (in a browser) w/ JSON Payloads

«uses to control MP3 Player»

User

5 Button Joystick
PA[1-5] Config: Input, w/Pull Down
High Speed, triggers on Rising Edge

Pot Used To Adjust Volume
Analog input on PA6 is Periodically Sampled via TIM4 Triggers TGRO every 10KHz

Green LED
Toggles Pin **PJ5** on each Frame R/W

Red LED
Toggles Pin **PJ13** every 500ms

SSD1306 OLED Display
I2C Addr: 0x78 (ChibiOS/HAL expect address to be shifted to the right by 1 bit)
Driver: I2CD1
I2C Pins:PD12 SCL w/ AF=4, PB7 SDA w/ AF=4 w/ (**open drain**)
PJ3: used for Reset on boot/started
PJ4: SA0 Pulled UP High

CODEC: wm8994
I2C Addr: 0x34 (ChibiOS/HAL expects Address to be shifted to the right by 1 bit)
Driver: I2CD4
I2C Pins :PB8 SCL w/ AF=4, PB9 SDA w/ AF=4 & (**open drain**)
SAI1 Pins: PE[4-6], PG7 w/ AF=16 (Push-Pull w/ Highest Speed)

SD Card Reader
Driver: SDCD2
SDC Pins: PG9=D0, PG10=D1,PB3=D2,PB4=D3,PD6=SCK, PD7=CMD
PushPull with **High Speed** w/ AF=11 for all except D2 & D3 AF 10

Headphone

Salah Mansour - Dec. 23rd, 2019