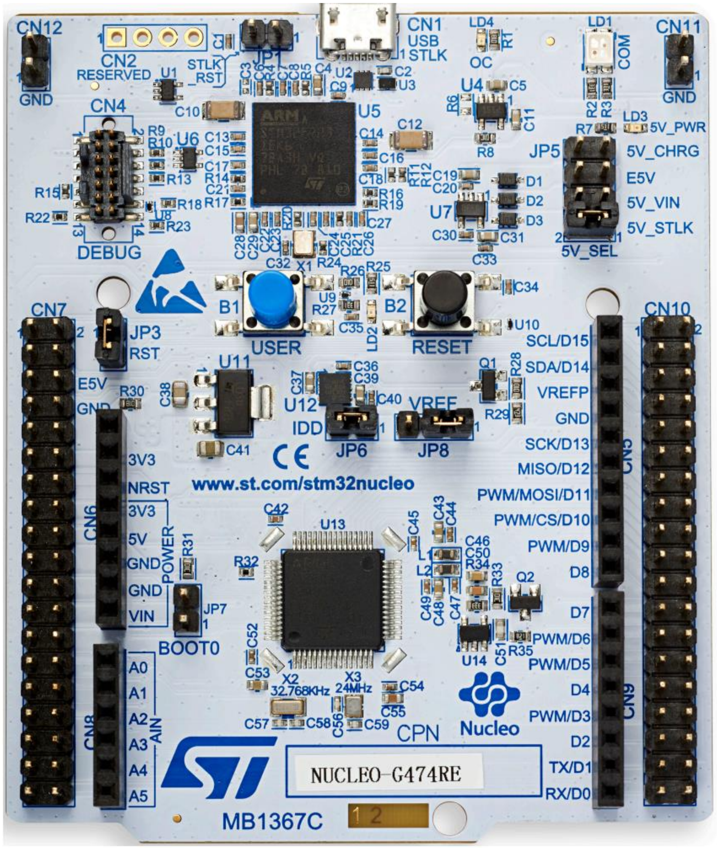
STM32G4 Technique Training

life.augmented

STM32G4 hands-on LAB3

STM32G4 hands-on

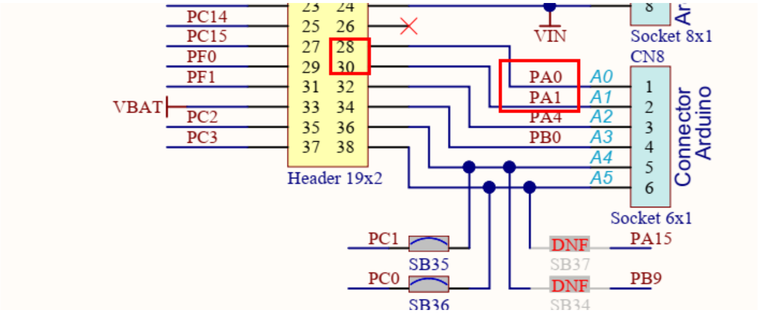
- Objective
  - ADC HW Oversampling
  - Enable DMA feature : get ADC result then transfer into user buffer
- Step 1
  - Create a project in STM32CubeIDE
    - Configure ADC1 input ch1 and ch2 + HW Oversampling
  - Generate initialization code
- Step 2
  - Configure DMA Channel
  - Generate initialization code
  - Add the user code
  - Compile again and run

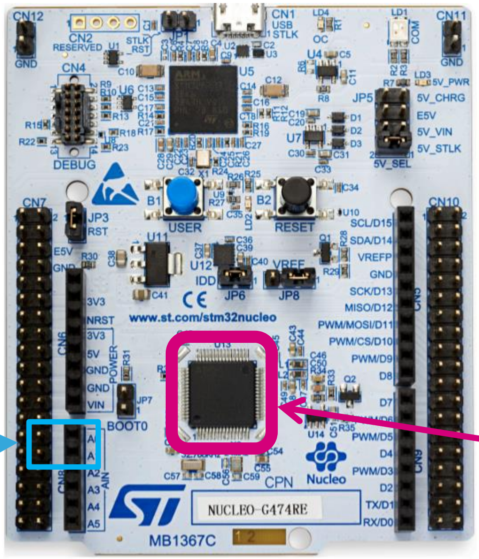


LAB2 ADC HW Oversampling + DMA

NUCLEO-G474RE Board

- MCU – **STM32G474RET6**



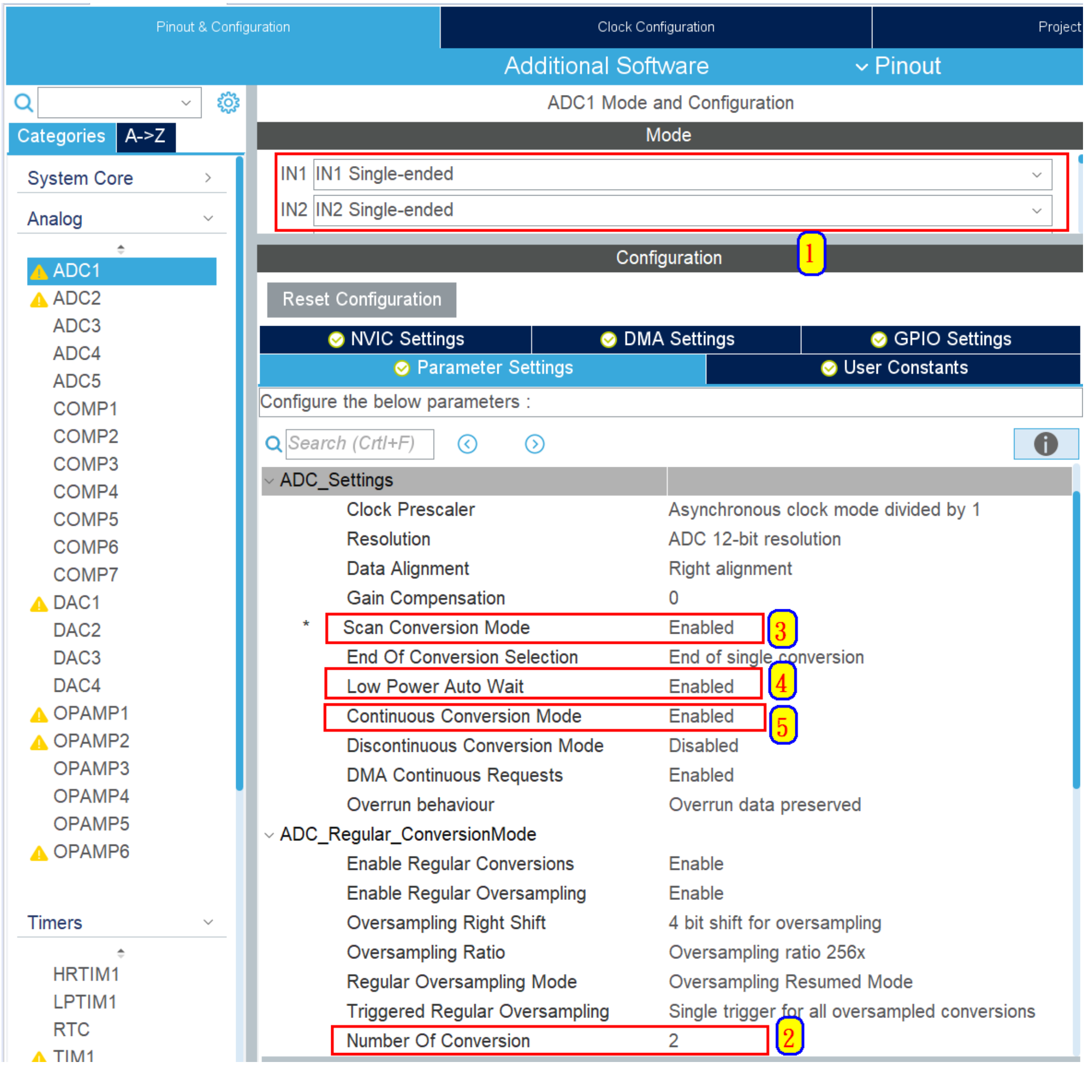


ADC1_IN0 and ADC1_IN1

STM32G4 MCU
- ARM Cortex-M4 core 170MHz
- 512KBytes of Flash
- 128 KB of RAM

Regenerating the initialization code

- Selector ADC1 for use
  - Configure ADC1 inupt ch1 and ch2 with Single-ended mode.
  - Set 2ch of Conversion.
  - Enabled Scan Conversion Mode.
  - Set Low-power modes (AUTDLY).
  - Enable Continuous Conversion Mode



ADC1 Mode and Configuration

Mode

IN1 IN1 Single-ended

IN2 IN2 Single-ended

Configuration

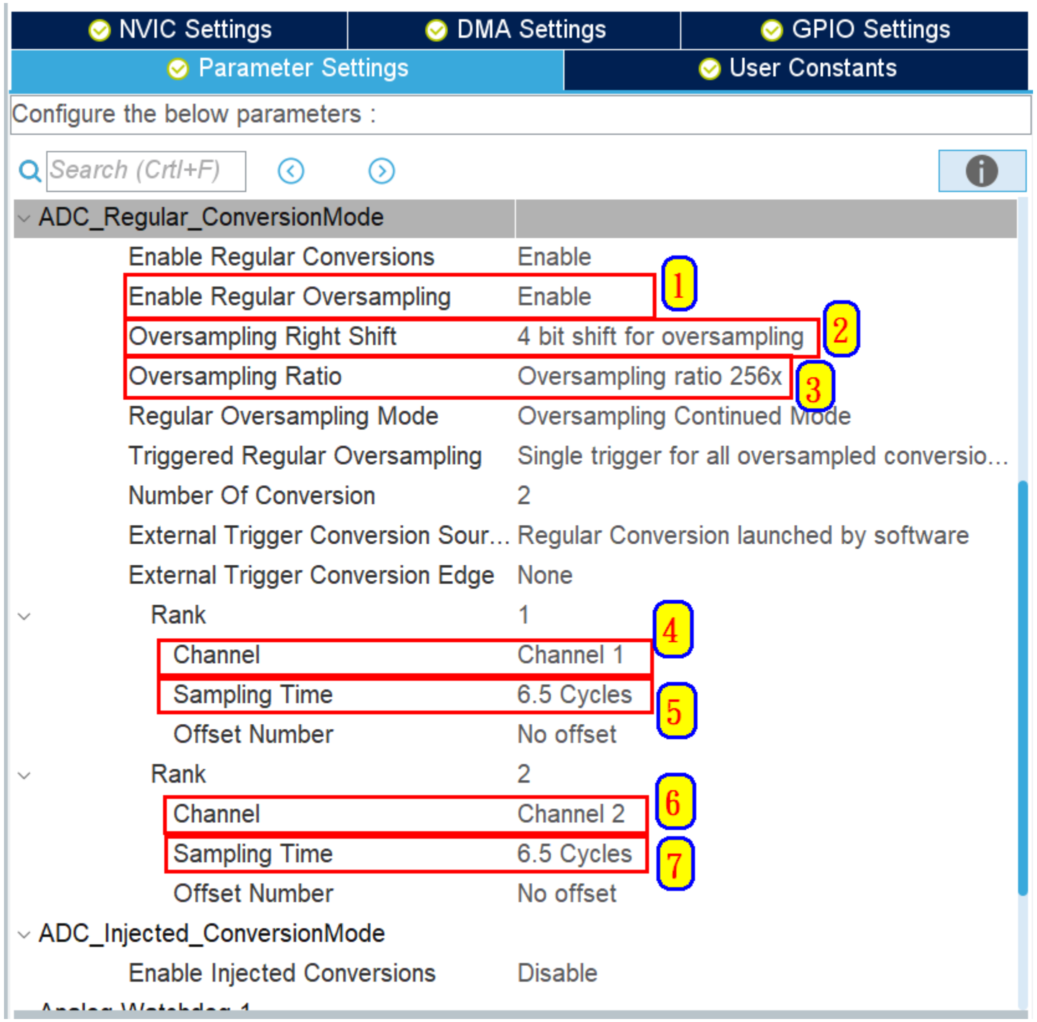
Configure the below parameters :

```
ADC_Settings
    Clock Prescaler                      Asynchronous clock mode divided by 1
    Resolution                           ADC 12-bit resolution
    Data Alignment                       Right alignment
    Gain Compensation                    0
  * Scan Conversion Mode                 Enabled
    End Of Conversion Selection          End of single conversion
    Low Power Auto Wait                  Enabled
    Continuous Conversion Mode           Enabled
    Discontinuous Conversion Mode        Disabled
    DMA Continuous Requests              Enabled
    Overrun behaviour                    Overrun data preserved
ADC_Regular_ConversionMode
    Enable Regular Conversions           Enable
    Enable Regular Oversampling          Enable
    Oversampling Right Shift             4 bit shift for oversampling
    Oversampling Ratio                   Oversampling ratio 256x
    Regular Oversampling Mode            Oversampling Resumed Mode
    Triggered Regular Oversampling       Single trigger for all oversampled conversions
    Number Of Conversion                 2
```

life.augmented

Regenerating the initialization code

- ADC_Regular_Conversion Mode
  - Enable Regular Oversampling.
  - 4 bit shift for oversampling.
  - Oversampling ratio 256x.
  - Rank1 for ch1 6.5 Cycles Sampling Time.
  - Rank2 for ch2 6.5 Cycles Sampling Time.



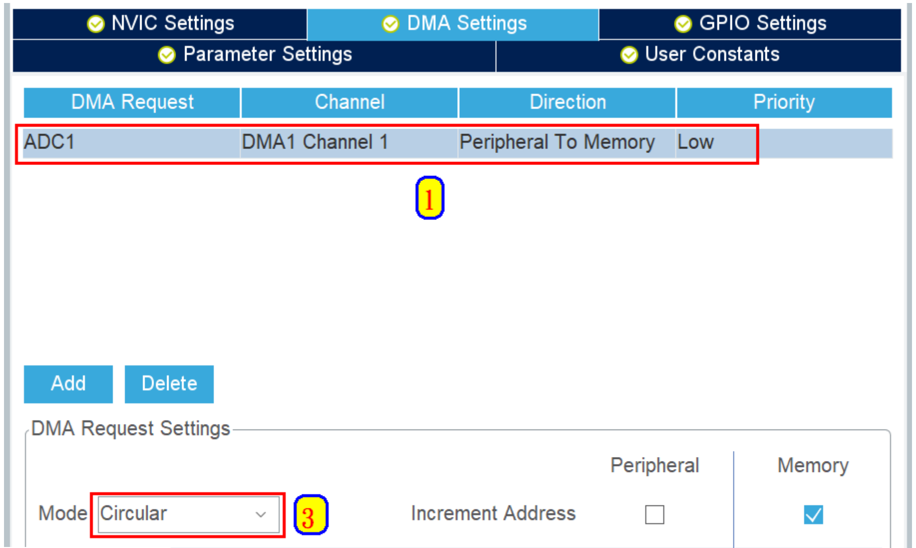
ADC_Regular_ConversionMode	
Enable Regular Conversions	Enable
Enable Regular Oversampling	Enable
Oversampling Right Shift	4 bit shift for oversampling
Oversampling Ratio	Oversampling ratio 256x
Regular Oversampling Mode	Oversampling Continued Mode
Triggered Regular Oversampling	Single trigger for all oversampled conversio...
Number Of Conversion	2
External Trigger Conversion Sour...	Regular Conversion launched by software
External Trigger Conversion Edge	None
Rank	1
Channel	Channel 1
Sampling Time	6.5 Cycles
Offset Number	No offset
Rank	2
Channel	Channel 2
Sampling Time	6.5 Cycles
Offset Number	No offset
ADC_Injected_ConversionMode	
Enable Injected Conversions	Disable
Analog Watchdog 1	

life.augmented

Regenerating the initialization code

- DMA Settings
  - Selector DMA Request of ADC1.
  - Word of Transfer Data Width.
  - DMA in circular mode to handle a continuous analog input data stream.

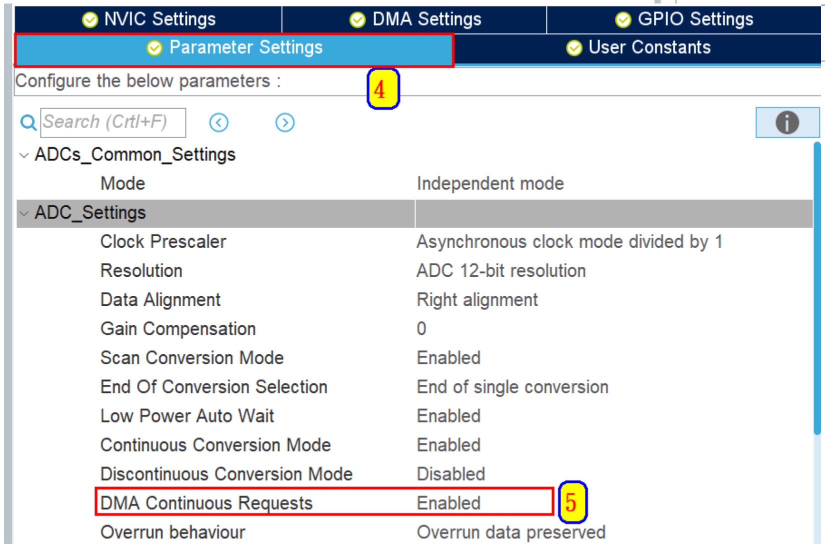
- Enabled DMA Continuous Requests
  - Go back Parameter Settings Tab to Setting



DMA Request	Channel	Direction	Priority
ADC1	DMA1 Channel 1	Peripheral To Memory	Low

DMA Request Settings

		Peripheral	Memory
Mode: Circular	Increment Address	□	☑
	Data Width	Word	Word

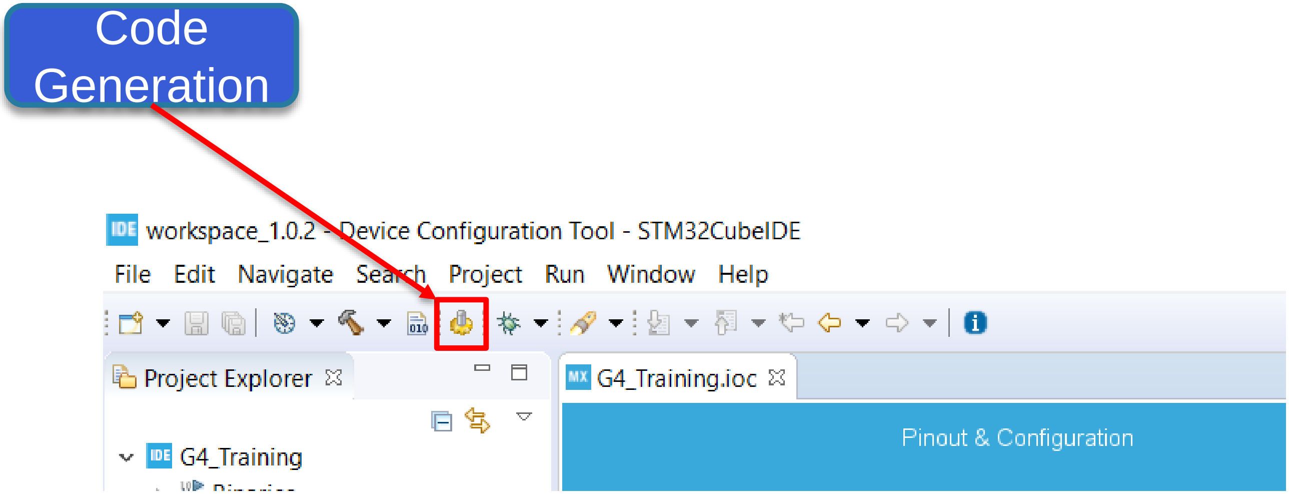


Configure the below parameters :

ADCs_Common_Settings	
Mode	Independent mode
ADC_Settings	
Clock Prescaler	Asynchronous clock mode divided by 1
Resolution	ADC 12-bit resolution
Data Alignment	Right alignment
Gain Compensation	0
Scan Conversion Mode	Enabled
End Of Conversion Selection	End of single conversion
Low Power Auto Wait	Enabled
Continuous Conversion Mode	Enabled
Discontinuous Conversion Mode	Disabled
DMA Continuous Requests	Enabled
Overrun behaviour	Overrun data preserved

Generate Code

- STM32CubeIDE
  - Click “Code Generation” button to generate source code.



Code Generation

Complete the new code

- Add user buffer in application

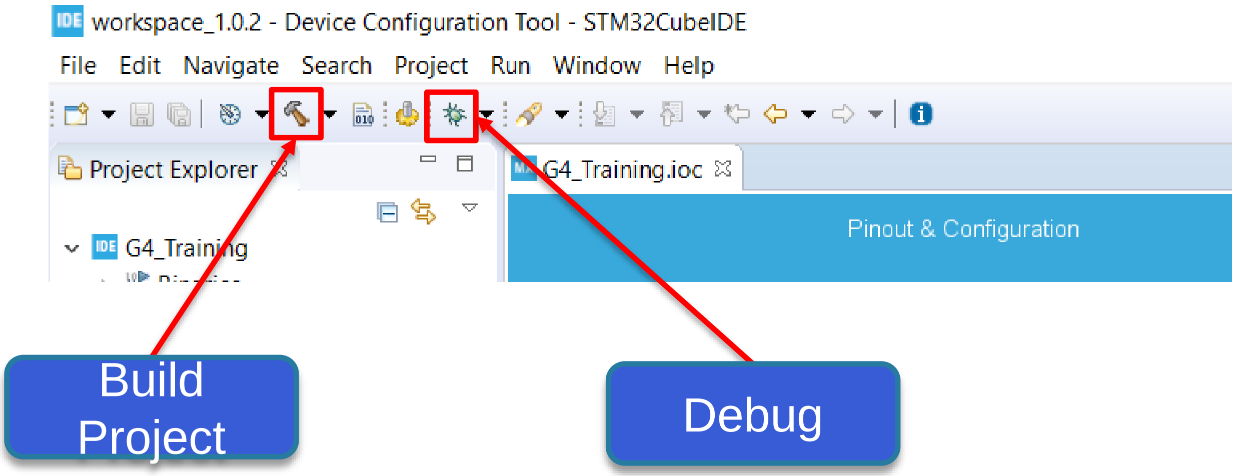
```
/* USER CODE BEGIN 0 */
__IO uint32_t aADCDualConvertedValue[2];
/* USER CODE END 0 */
```

- Add application code in porjcet
  - Open main.c then add HAL_ADCEx_Calibration_Start( ) and HAL_ADC_Start_DMA( )API in Infinite loop.
  - Each ADC provides an automatic calibration procedure
  - ADC generates a DMA transfer request each time a new conversion data is available in the data register

```
/* USER CODE BEGIN 2 */
 HAL_ADCEx_Calibration_Start(&hadc1, ADC_SINGLE_ENDED);
 HAL_ADC_Start_DMA(&hadc1, (uint32_t *)aADCDualConvertedValue,\
                                    sizeof(aADCDualConvertedValue)/sizeof(uint32_t));
 /* USER CODE END 2 */
```

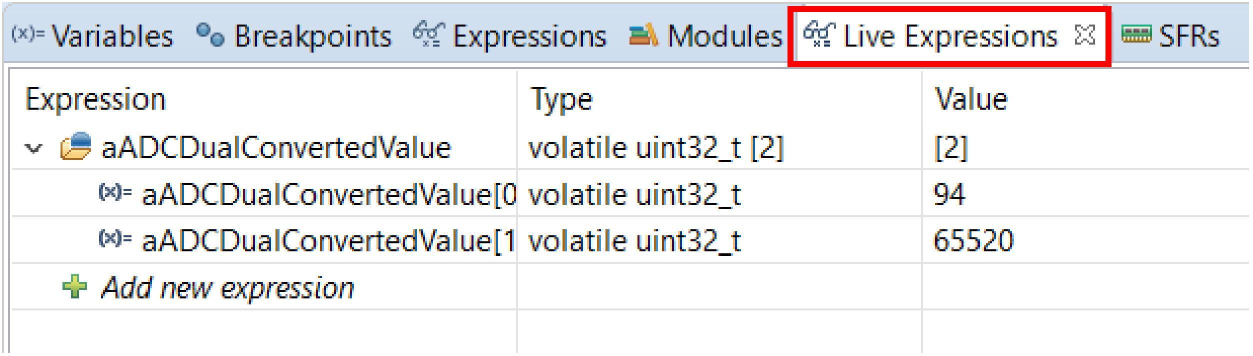
Build and Debug project

- STM32CubeIDE
  - Build the project by click “Make” and then “Download and Debug”.



Life Watch

- Add user buffer array in Live Expressions Field



Expression	Type	Value
aADCDualConvertedValue	volatile uint32_t [2]	[2]
aADCDualConvertedValue[0	volatile uint32_t	94
aADCDualConvertedValue[1	volatile uint32_t	65520
Add new expression		

Releasing Your Creativity

/STM32 @ST_World community.st.com stm32g4-online-training

www.st.com/STM32G4

life.augmented