# Features of project

1. Stm communicates with other stm using uart (UART)
2. GP Timer will send data in a regular intervel
3. Sends Phot-registers reading (ADC + UART)
4. Send update to the computer or log file (UART)
5. Receive data from other stm and response by setting light state (UART)
6. Reveive data from computer and turns on led or off (UART)
7. Timer: a particular data will be sent in a fixed delay.

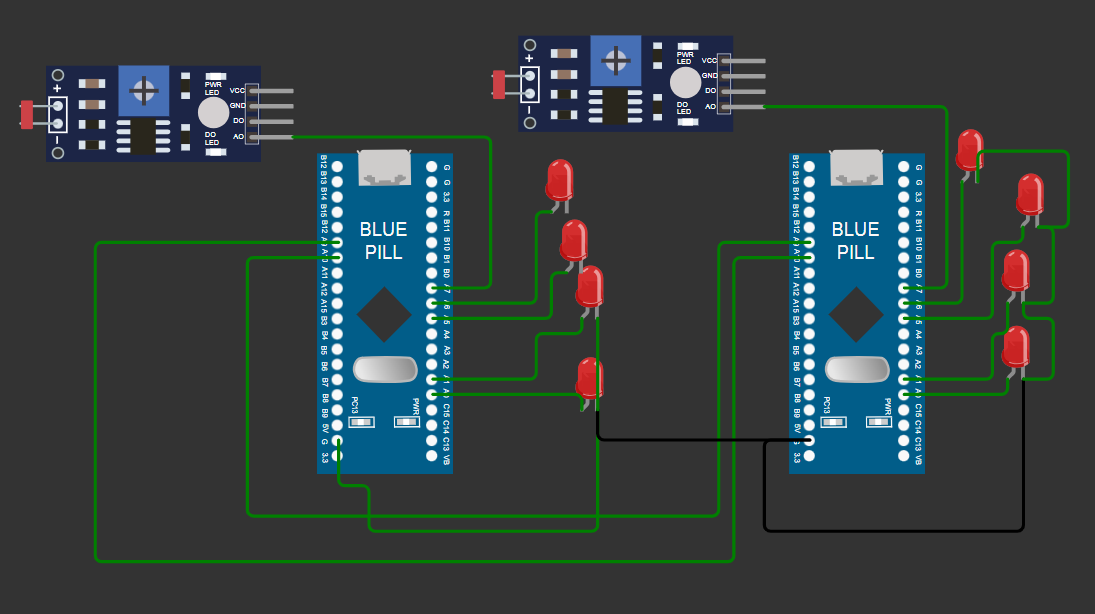
# Features of stm32 used in project

1. UART
2. ADC
3. Systick timer
4. Time gpio trigger
5. All in interrupt

# Project functions

1. Function for clock
2. Function for pin config
3. Function for systick setup
4. Function for time delay
5. Function for UART enable (both uart 1 and 2)
6. Function for ADC
7. Timer config function
8. All interrupt vector function
9. Main function.

# Circuit diagram:

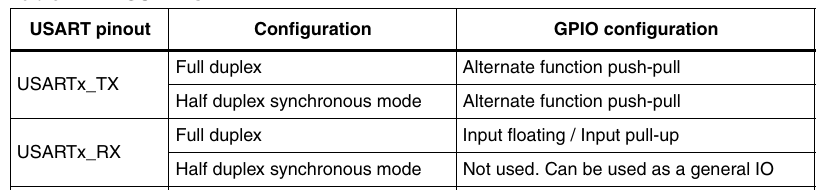


# Clock enable

* APB1ENR UART2, Timer2
* APB2ENR UART1, Systick, PA,ADC, AFIO

# Pin config:

## For UART:



* PA9: UART1 Tx : Output AF push-pull
* PA10: UART1 Rx : Input Floating
* PA2: UART2: Tx output AF push-pull
* PA3: UART2 Rx: input Floating
* PA0: TX signal for uart1 : GPIO output pushpull
* PA1: RX signal for uart1 : GPIO output pushpull
* PA4: Response of received data from USART1 : GPIO OUTput pushpull
* PA5: TxRx signal for UART2: GPIO output push pull
* pA6: Rx signal for UART2: GPIO output push pull
* PA7: Response of received data from USART2 : GPIO output pushpull
* PB0: Analog input for ADC : 0000

# SysTick Config:

1. SysTick Load: 72000-1;
2. Clock source cpu

# Timer setup:

1. Set counter value 0 in TIM2->CNT
2. Set the TIM2-> PSC value. CK\_CNT = fCK\_PSC / (PSC[15:0] + 1). (for 1ms, 7200-1)
3. Load autoreload register TIM2->ARR
4. Enable Update Interrupt in TIM2->DIER bit 0;
5. Set count Direction in TIM->2CR1 bit 4. Set is downcounter
6. Enable Timer in TIM2->CR1 bit 0;
7. Write interrupt vector runtime.

# UART setup ( 1 and 2 )

1. IN CR1:
   1. Enable RE in bit 2
   2. Enable TE in bit 3
   3. Eable RXNEIE (Rx not empty interrupt enable) in bit 5
   4. Eable UE( USART enable) in bit 13
2. Set baud rate in USART->BRR
3. Config in , Tx as AF output, RX as GPIO input push-pull
4. Enabe UART interrupt vector (NVIC)
5. Load data to USART->DR to start transmission
6. When data is received, RXNE will be set in SR and an interrupt will be call call. Write code in interrupt handeler.

# ADC setup:

In CR1:

1. Enable Analog watch dog on bit 23.
2. Enable Analog watch dog Interrupt on bit 6
3. Select analog watchdog channel in bit 0 to 4 : For pin B0, channel is 8, and for channel 8 , CR1[4:0] value is 01000

In CR2:

1. Enable continuous conversion in Bit 1.
2. Enable ADC on bit 0 . \*\*\* 2nd time bit set is required. Details on manual

In ADC\_HTR:

1. Give high threshold value for watchdog interrupt

In ADC\_LTR:

1. Give low threshold value for watchdog interrupt.

ADC\_DR holds the converted value.

In SQR3, put the channel number of Analog pin (for B0 , channel number is 8).

**//before ADON, cont and sqr value must be configure.**

**In Interrupt handling:**

**AWD flag in SR must be clear at the end of the handing function, otherwise the function will be a dead loop. Because as soon as flag is cleared, another interrupt can be occour before leaving the handler function.**

**Why clear AWD flag after handling?**

* **The AWD flag indicates a threshold violation.**
* **Clearing it before handling may cause the event to be missed or mishandled.**
* **Best practice: handle the event first, then clear the flag to preserve the cause of the interrupt.**