Shammah Thao

EEE 174 - CpE 185 Lab Section #2

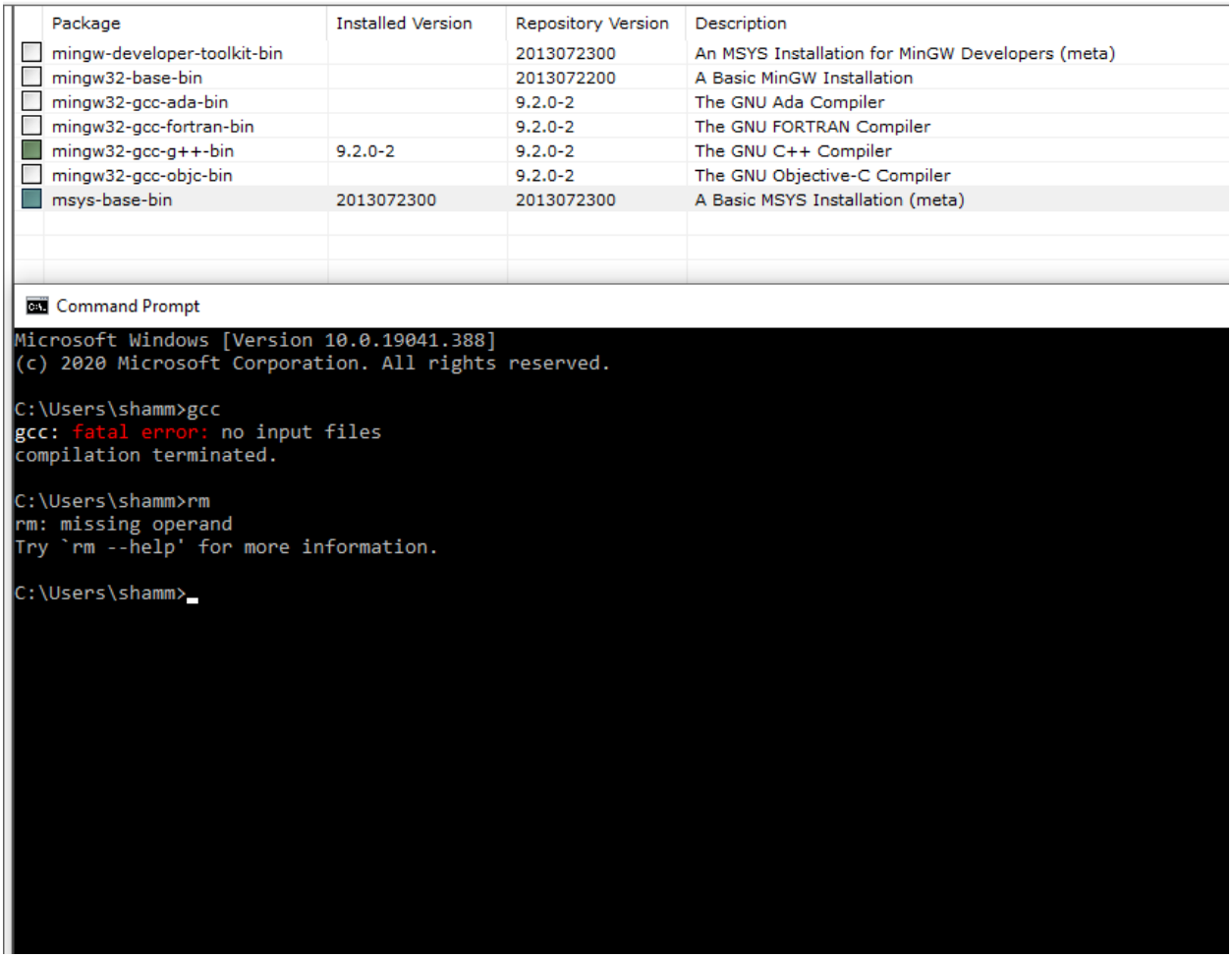
Monday & Wednesday

Lab 0 - Introduction to Microcontrollers and Lab Tools

Dahlquist

Part 0:

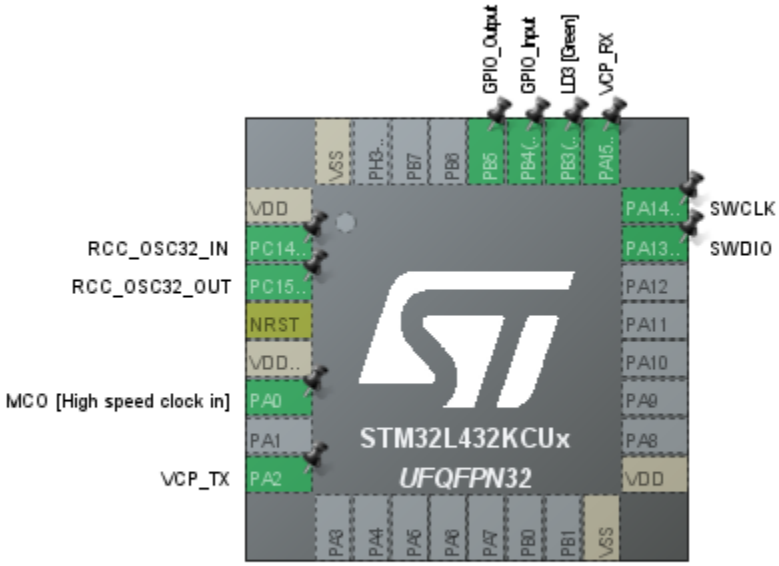
To start this lab, since we are not using atomic, we first have to install minGW, selecting mingw32-gcc-g++-bin and msys-base-pin to be installed into the computer.



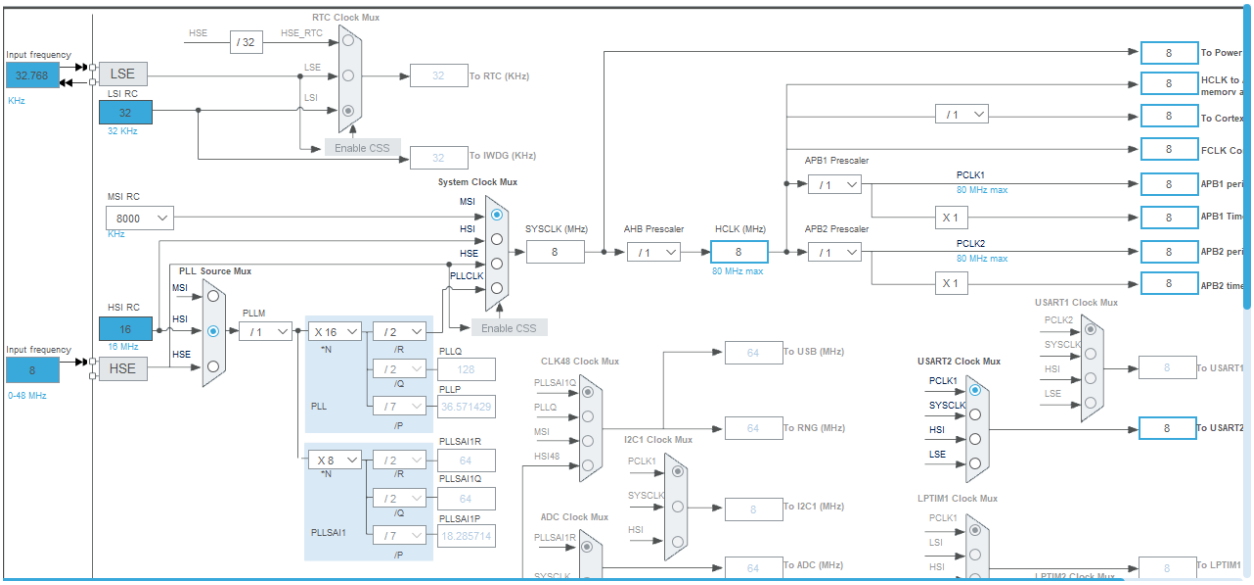
Package	Installed Version	Repository Version	Description
mingw-developer-toolkit-bin		2013072300	An MSYS Installation for MinGW Developers (meta)
mingw32-base-bin		2013072200	A Basic MinGW Installation
mingw32-gcc-ada-bin		9.2.0-2	The GNU Ada Compiler
mingw32-gcc-fortran-bin		9.2.0-2	The GNU FORTRAN Compiler
mingw32-gcc-g++-bin	9.2.0-2	9.2.0-2	The GNU C++ Compiler
mingw32-gcc-objc-bin		9.2.0-2	The GNU Objective-C Compiler
msys-base-bin	2013072300	2013072300	A Basic MSYS Installation (meta)

Part 1: General Purpose Input / Output

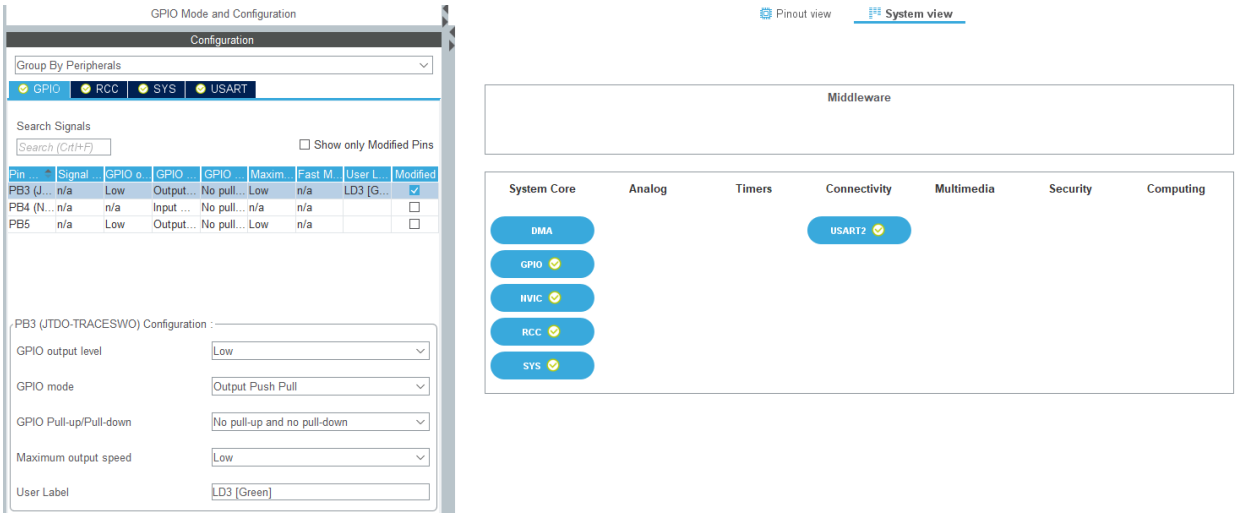
For this part, we had to open the stm ide software, where we have to make a new project and select our board that we are using for this lab.



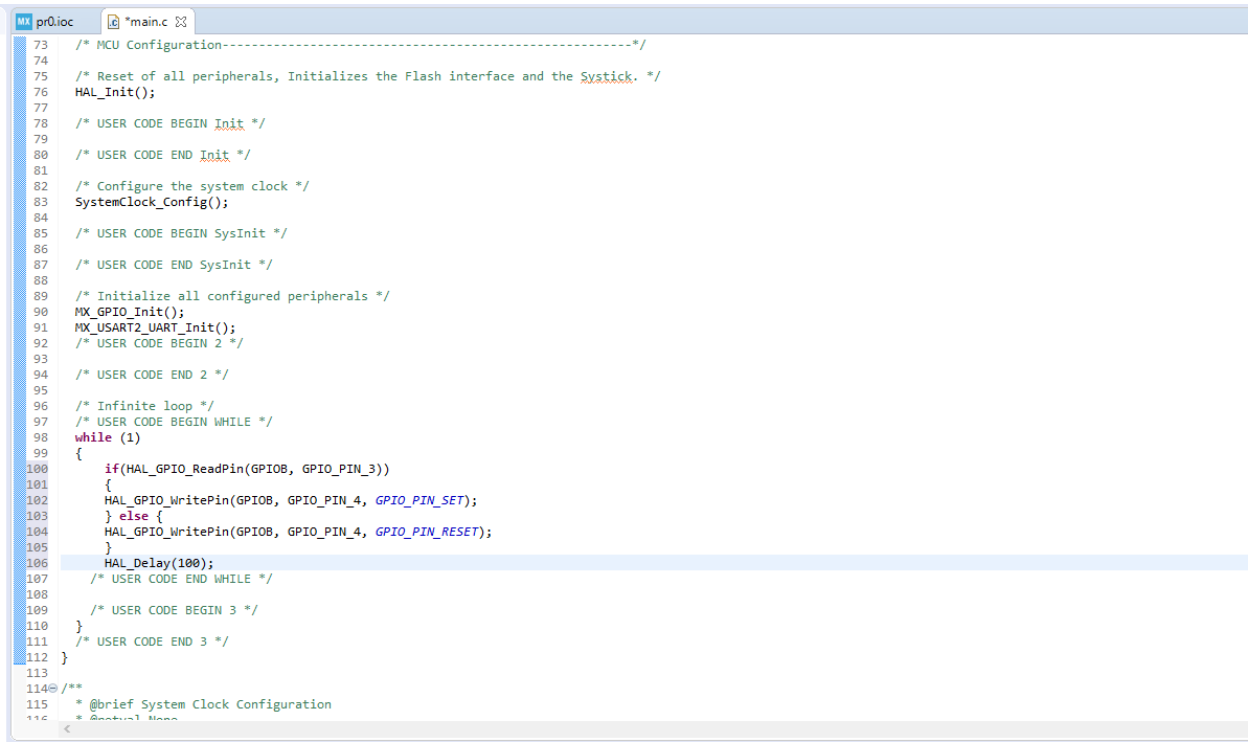
When it come up, we are to make our pin layout match to the one in the instruction paper.



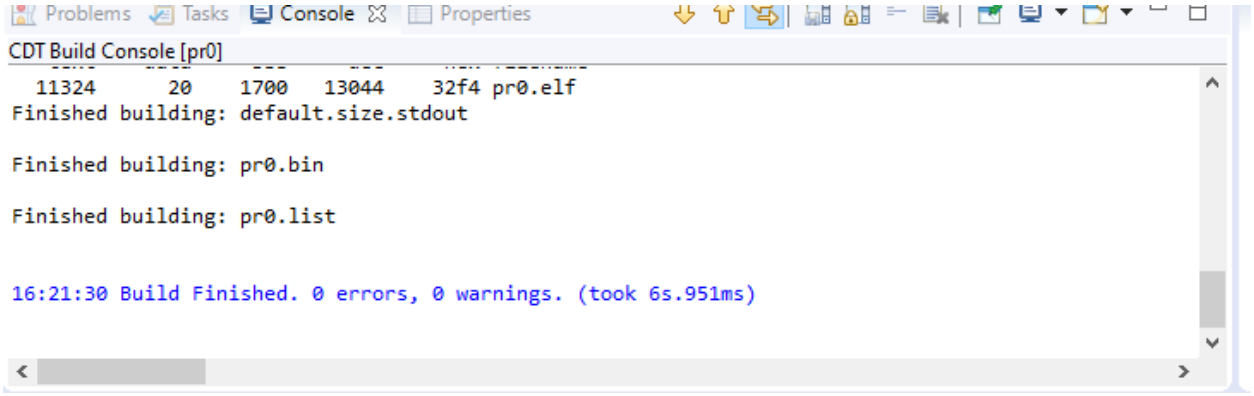
We then also have to match the clock configuration with the example, since the clock configuration outline was a little bit different, I just had to make it so all the number on the right is getting 8.



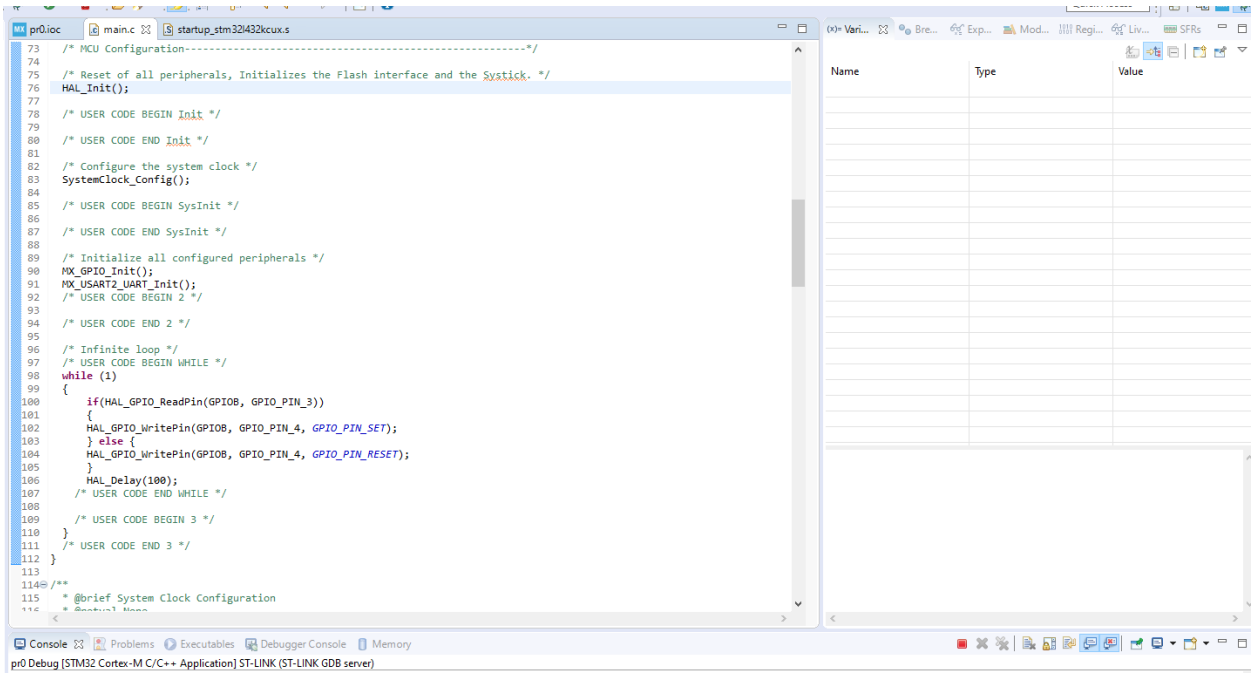
I then check if all the GPIO setting is like the given one.



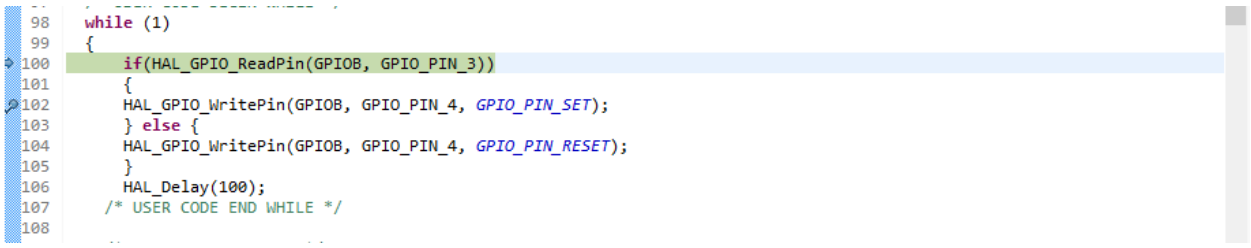
Then using the code that was given to use we copied and pasted it into the while loop, what this code did was that it made the LED light up when we pressed on the push button, after setting up the breadboard.



Building the project made it so it tell me if there is an error with my code.

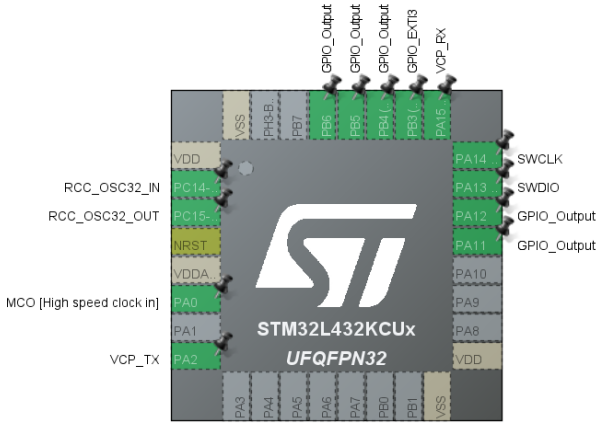


Then we learn how to use the debugger to go through the code step by step.



Created a breakpoint at 103 to start debugging from there.

After the debugging process we then start messing around with the interrupt of this software. We first had to change the pin PB3 to GPIO_EXIT3.



EXTI line3 interrupt	0	0

Then we activated the interrupt from the NVIC

```
void EXTI3_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI3_IRQn 0 */
     HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_4); //Toggle the state of pin PC9
  /* USER CODE END EXTI3_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_3);
  /* USER CODE BEGIN EXTI3_IRQn 1 */
  HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_5);
  HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_6);
  /* USER CODE END EXTI3_IRQn 1 */
}
```

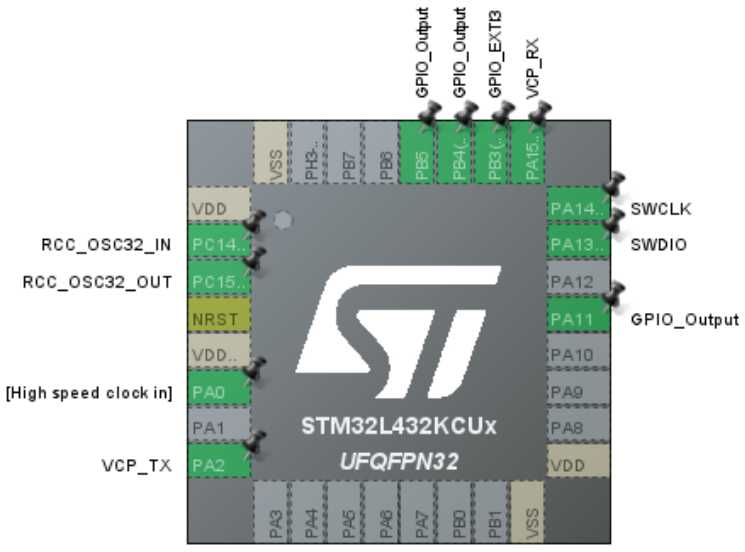
We used the code given to use and put it into the stm32l4xx_it.c file, it is what helps interrupts the code. After putting this code in and running it. We got that once the button is press it stay solid, until it was pressed again to stop it. The interrupts prevent the LED to be turn off till it was told to.

DEMO:

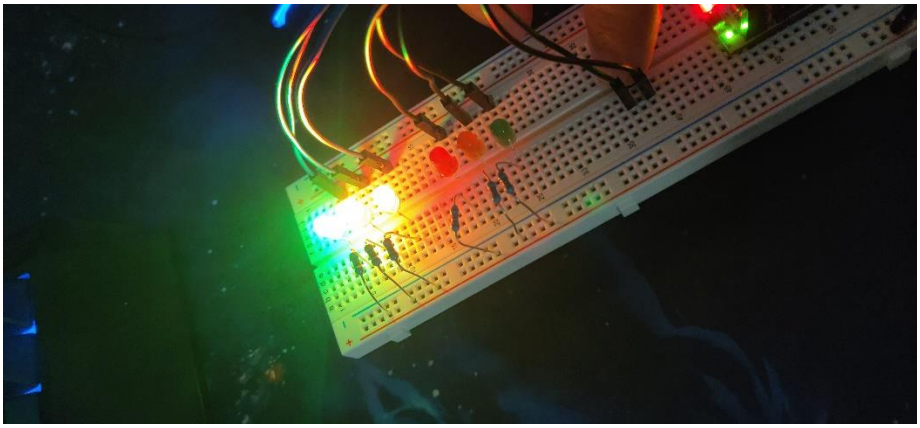
For the demo we had to add 2 more led, and make it blink in a random order. For the code I made the LED blink randomly.

```
  while (1)
  {
    /* USER CODE END WHILE */
      if(HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_3)){
         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_SET);
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_11, GPIO_PIN_SET);
         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, GPIO_PIN_SET);
         HAL_Delay(500);
         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_SET);
         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, GPIO_PIN_SET);
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_11, GPIO_PIN_RESET);
         HAL_Delay(500);
         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_SET);
         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, GPIO_PIN_RESET);
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_11, GPIO_PIN_SET);
         HAL_Delay(500);
         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_RESET);
         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, GPIO_PIN_SET);
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_11, GPIO_PIN_SET);
          }else{
         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_RESET);
         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, GPIO_PIN_RESET);
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_11, GPIO_PIN_RESET);
          }
          HAL_Delay(500);
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

Using the pin layout below, I was able to set up my board to make this program run.

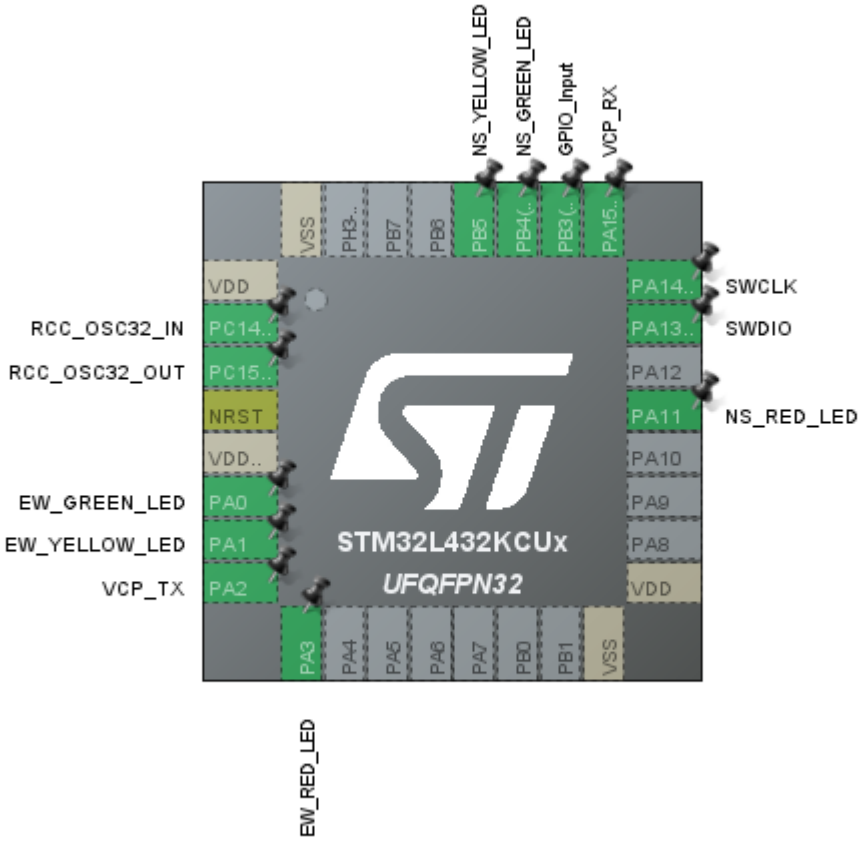


When the button is pressed and hold, the LED would flash in different combination until it is released.



Part 2: Finite State Machines Software Design Pattern

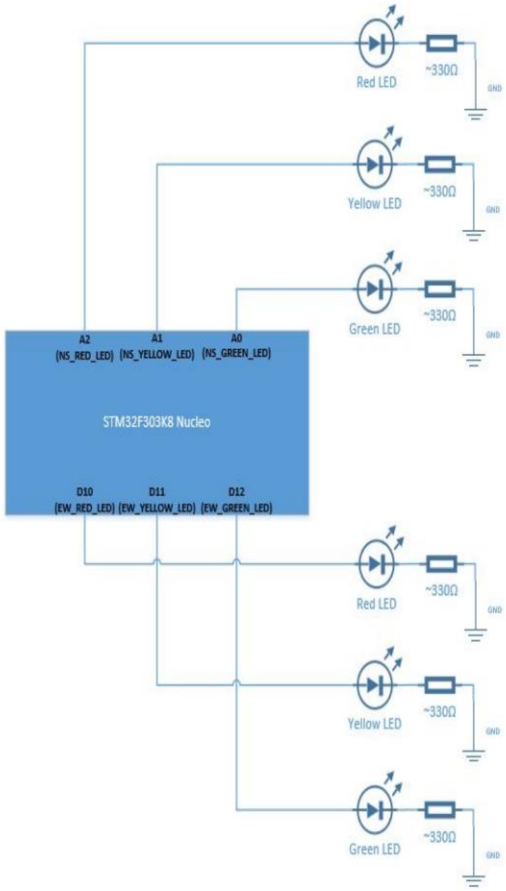
For Part 2, we had to make something like a stop light. For this part, we had to modify the pin layout to be like the own shown in the instruction.



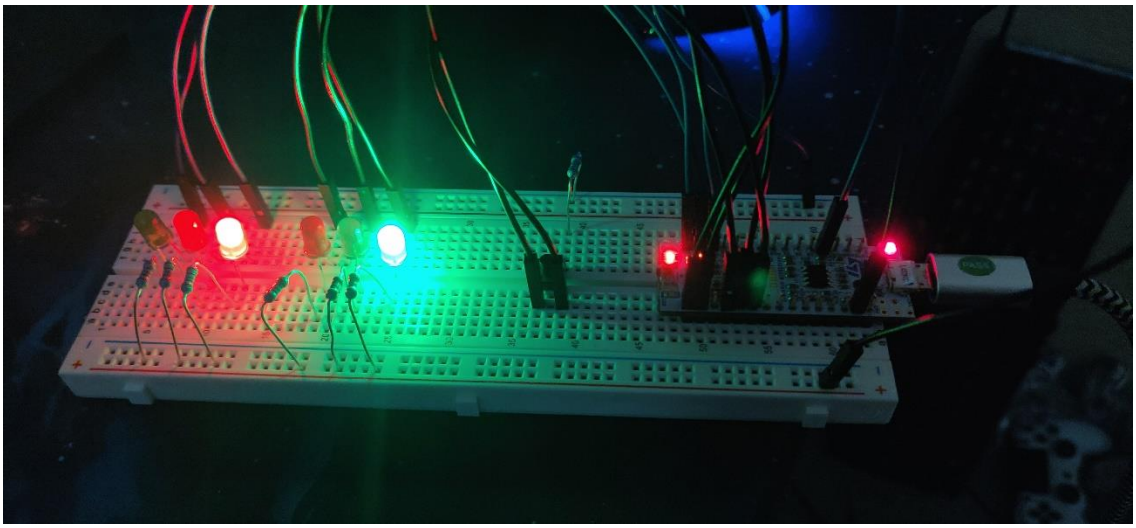
We then used the code that was given to use and modify it so it would be stop the lights at a red and green light. When the button is pressed it would continue, but if the button isn't pressed it would stay constantly at where it was at.

```
eSystemState EastWestPassHandler(void)
{
    HAL_GPIO_WritePin(GPIOB, NS_GREEN_LED_Pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOB, NS_YELLOW_LED_Pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOA, NS_RED_LED_Pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOA, EW_GREEN_LED_Pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOA, EW_YELLOW_LED_Pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOA, EW_RED_LED_Pin, GPIO_PIN_RESET);
    HAL_Delay(10*1000); //10 seconds
    if(HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_3)){
        return Transition_EW_State;
    }
    else
    {
        return NS_Stop_EW_Pass_State;
    }
    //return Transition_EW_State;
}
```

Using the diagram below we built the breadboard, along with modification which involves us adding a pushbutton onto it.



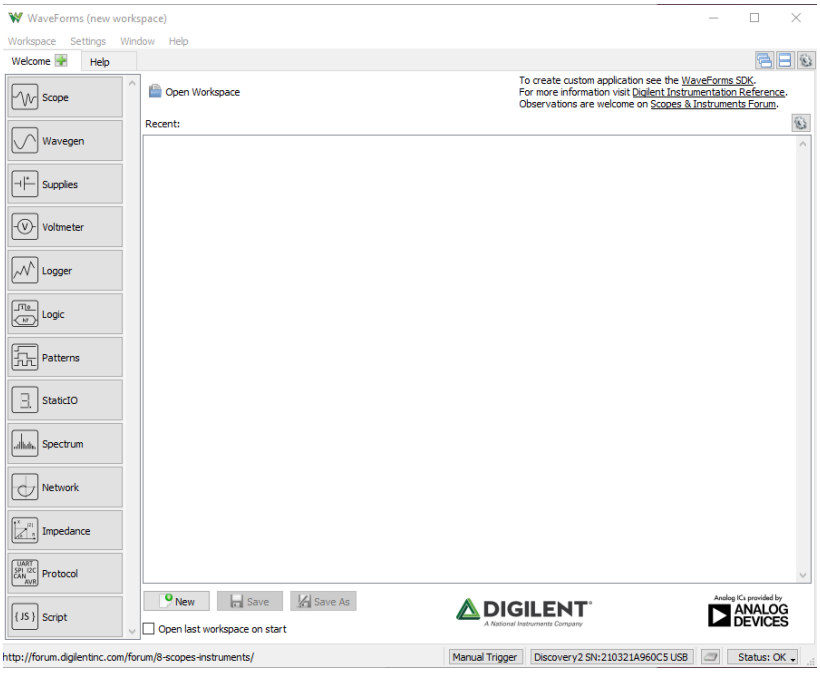
Using what was given to use along with the modification, I was able to make a working traffic light that would stop at a red and green and when the button is press it would run through again.



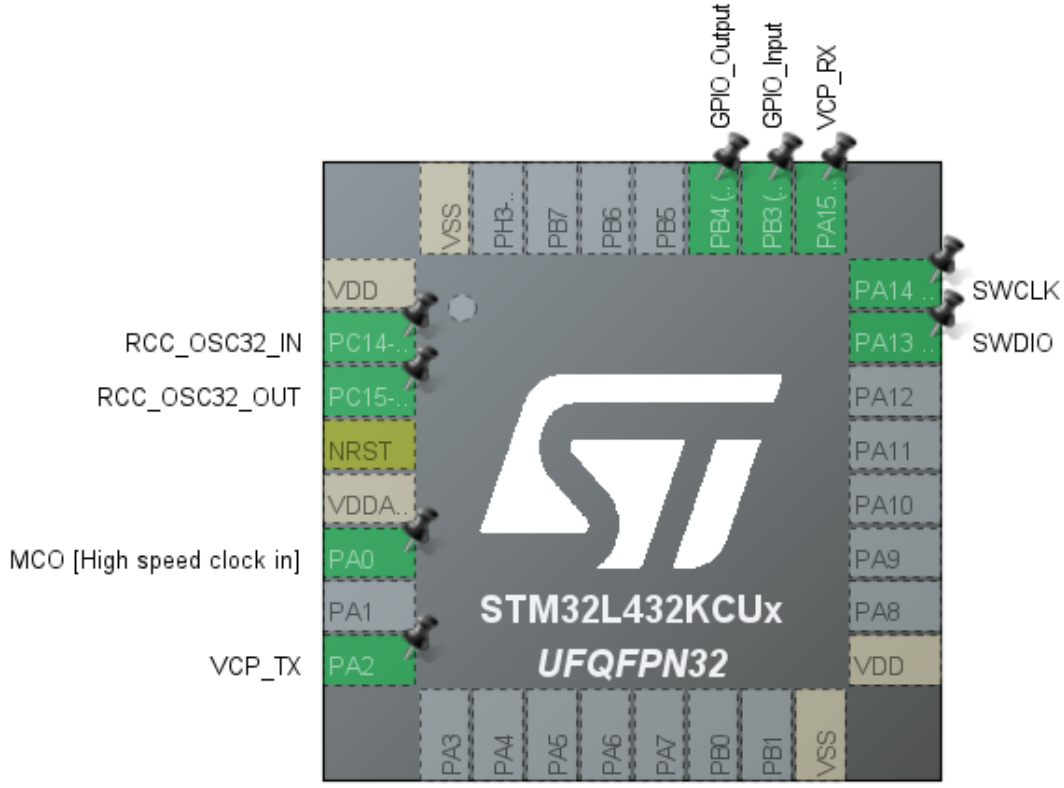
Part 3 : Analog Discovery: Extra Credit

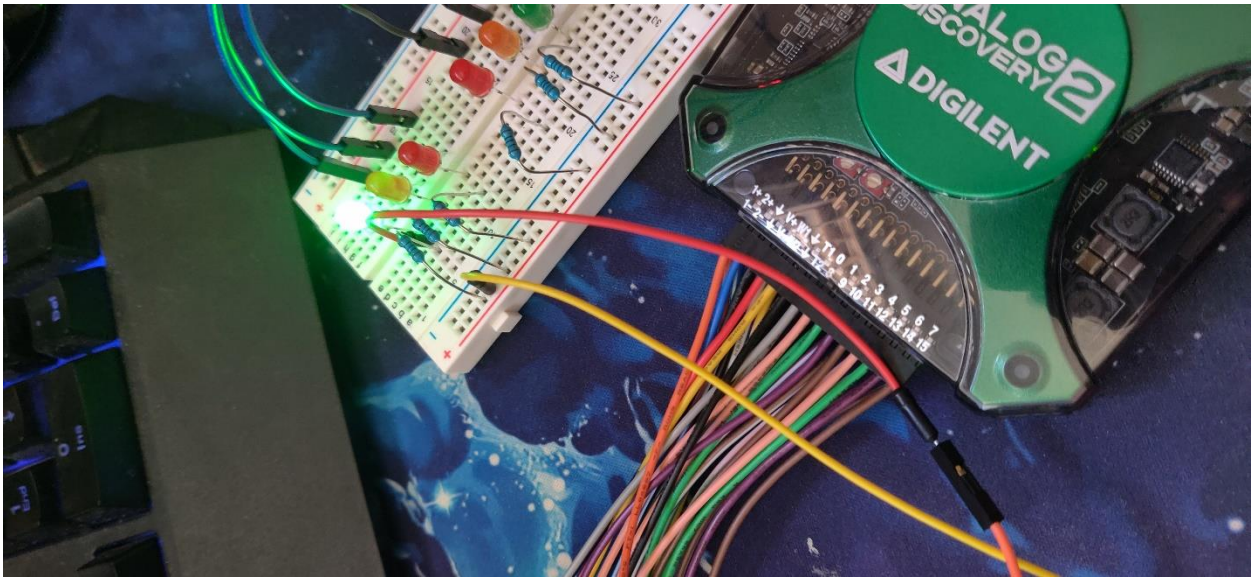
A) Introduction

For this part, we got got to connect the anonlog discovery along with our circuit on the breadboard. We then ran the first program from lab 1 on waveform.



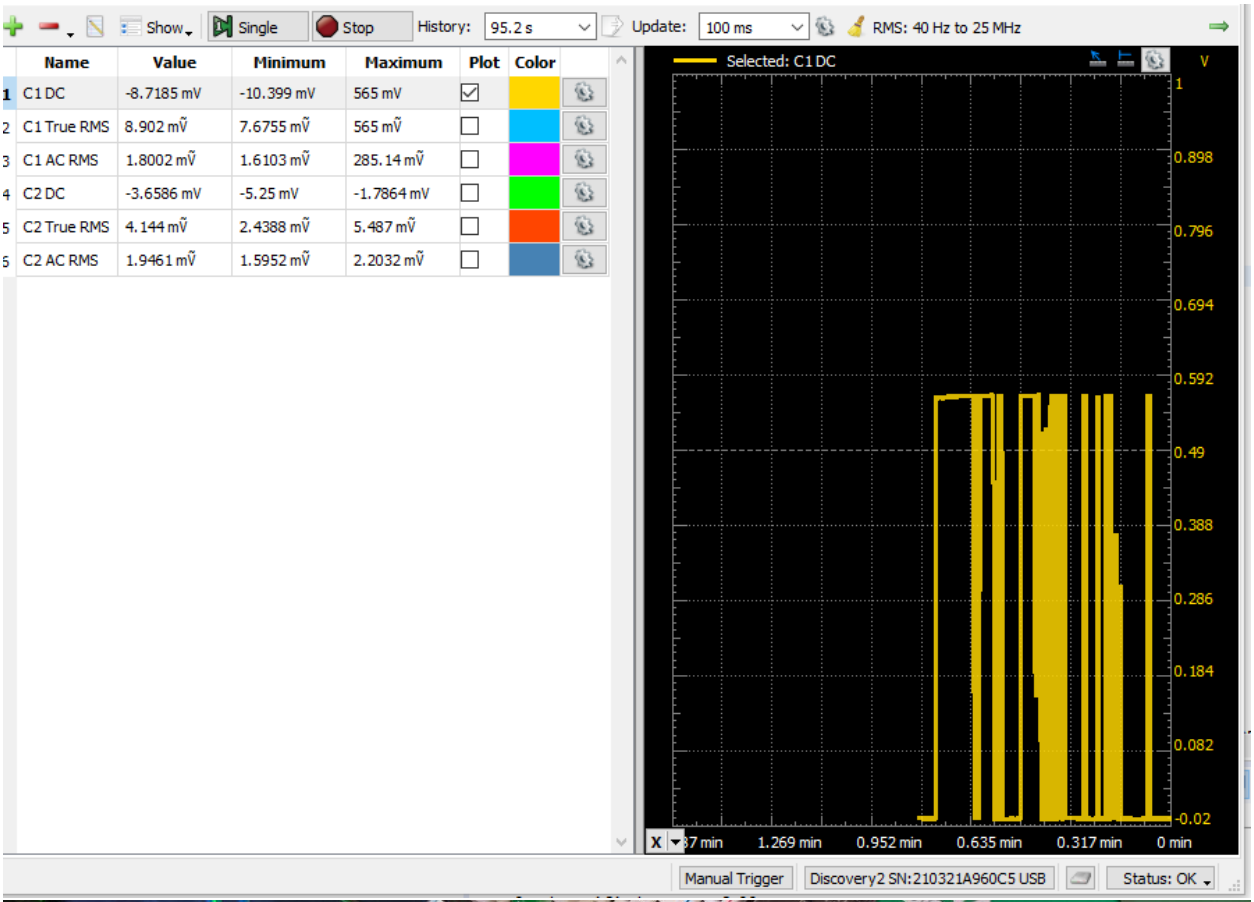
I did the pin as the original pin modules, as from part 1 of this lab.

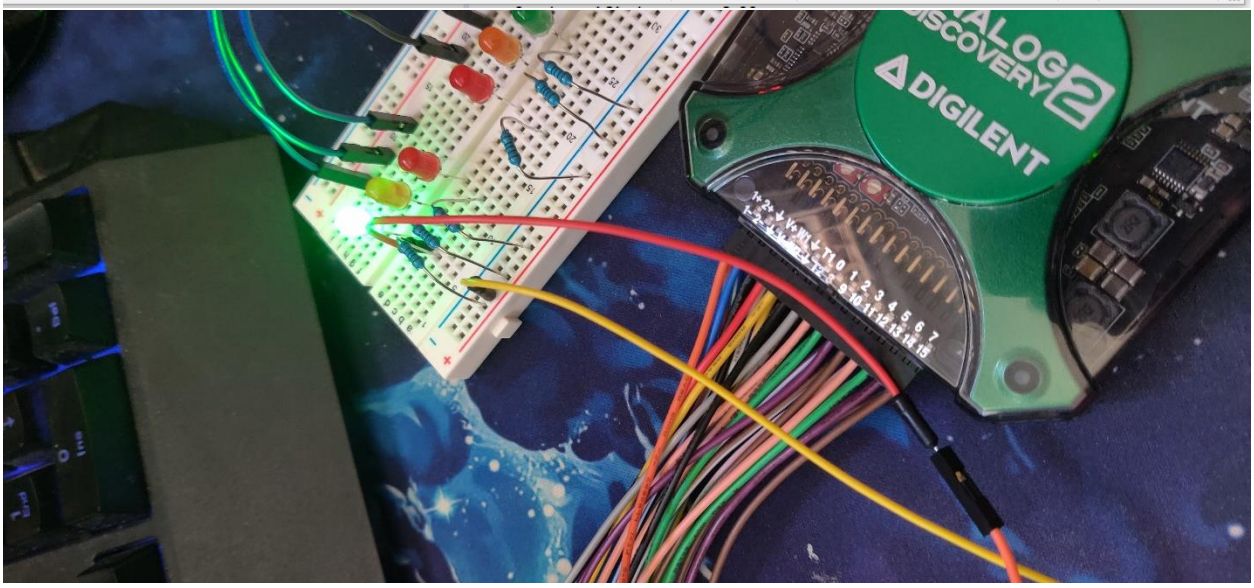

GPIO_Output
GPIO_Input
VCP_RX
VSS
PH3-..
PB7
PB6
PB5
PB4 (..
PB3 (..
PA15 ..
VDD
PA14 ..
SWCLK
RCC_OSC32_IN
PC14-..
PA13 ..
SWDIO
RCC_OSC32_OUT
PC15-..
PA12
NRST
PA11
VDDA..
PA10
MCO [High speed clock in]
PA0
PA9
PA1
STM32L432KCUx
PA8
VCP_TX
PA2
UFQFPN32
VDD
PA3
PA4
PA5
PA6
PA7
PB0
PB1
VSS


DIGILENT

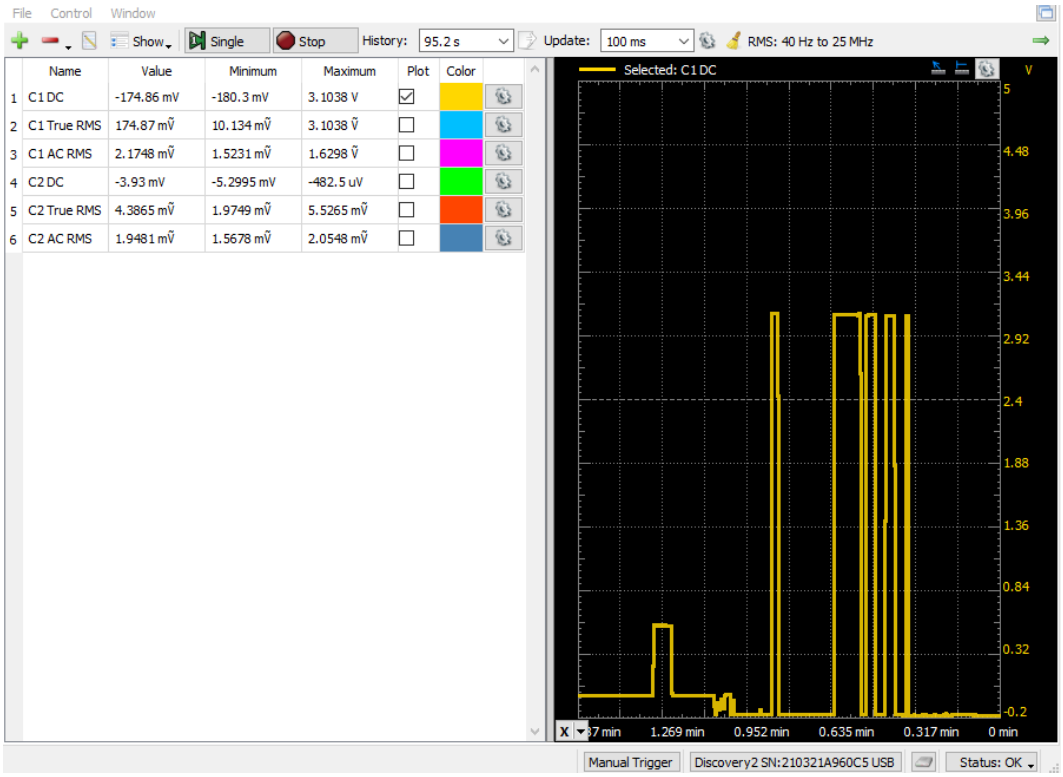
B) Logger(Voltmeter)

Running the logger on waveform, I got a constantly line when I pressed the button and when I didn't press the button there was no signal or power, rapidly pressing the button got me a chunk of yellow line stuck together.



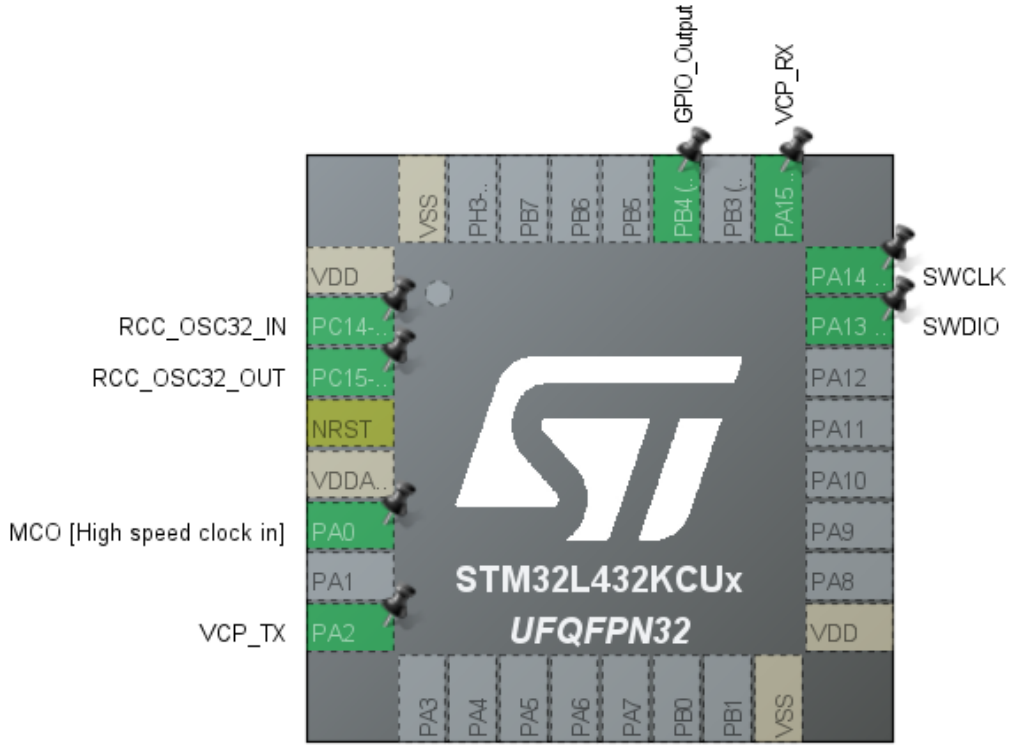


For this part I put the solid orange on to the other side of the LED and we got a different maximum. The maximum input the voltage got increase by a whole voltage.



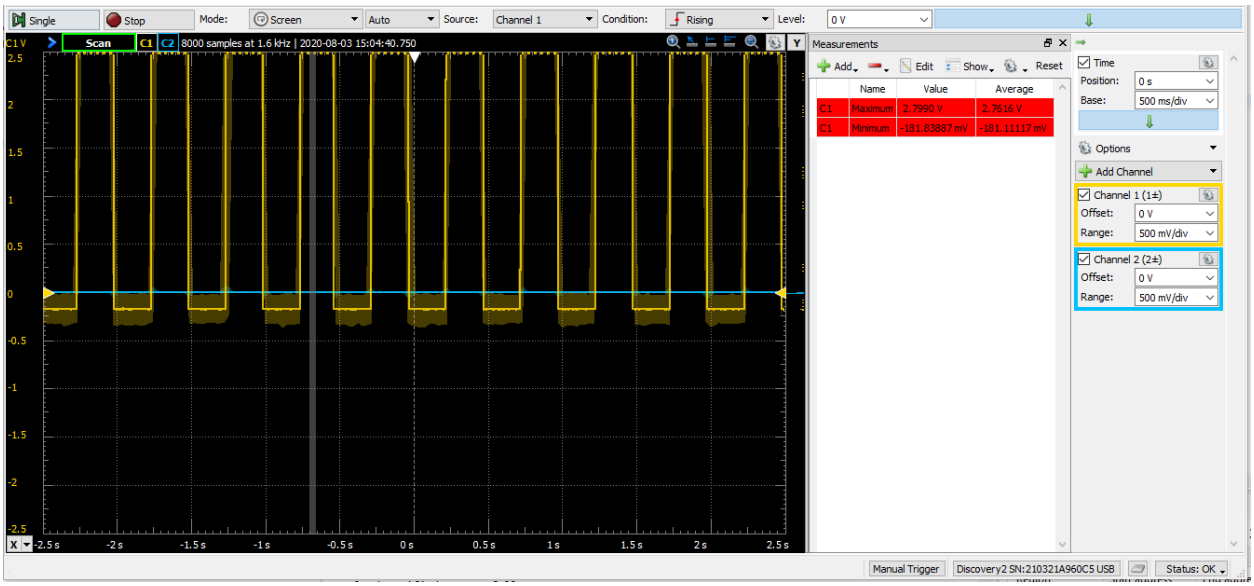
C) Oscilloscope

For this part, we must put in a code that we were given into the while loop in the main method. We also had to get rid of the push button on our pin modules.



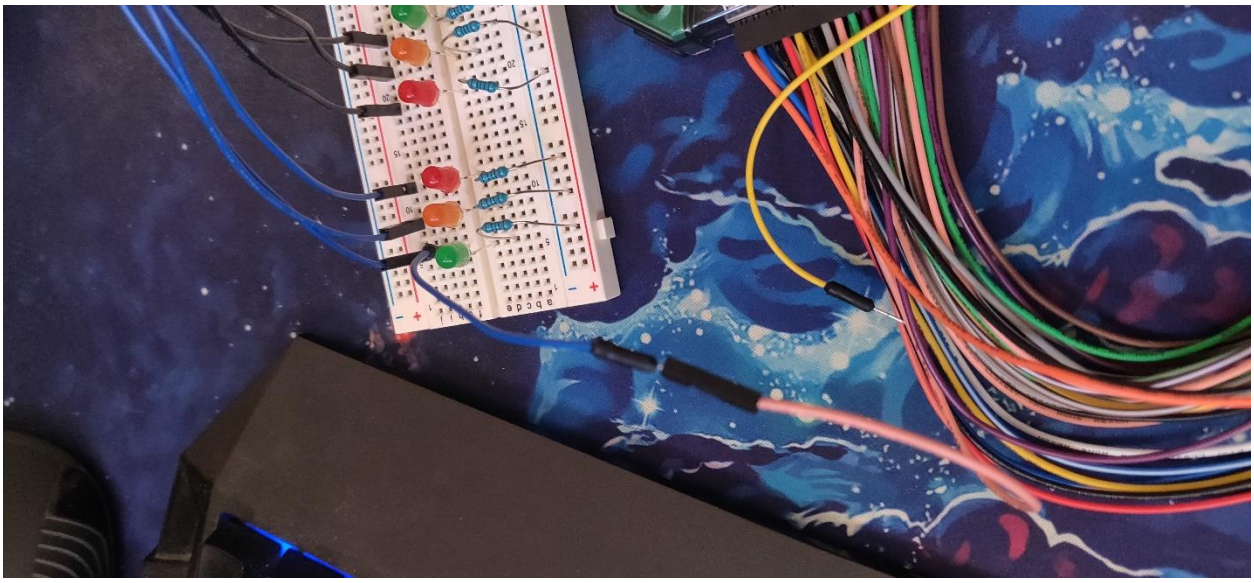

```
  while (1)
  {
    /* USER CODE END WHILE */
      while (1)
       {
       /* USER CODE END WHILE */
       HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_4); //Toggle the state of pin PC9
       HAL_Delay(250); //delay 250ms
       /* USER CODE BEGIN 3 */
       }

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

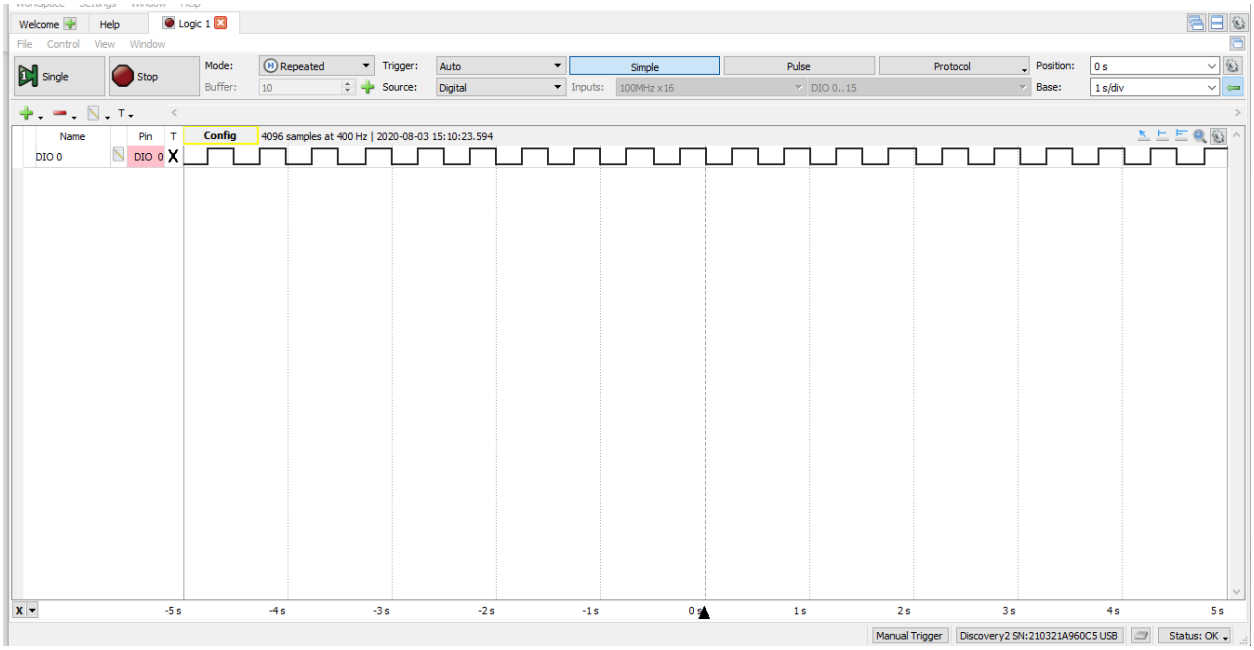


D) Logic Analyzer

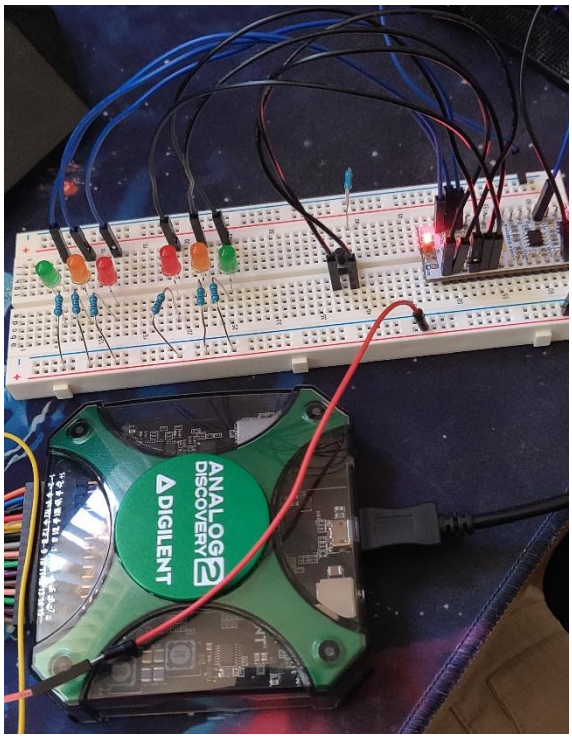
We connected the pink led to the anode side of the led.



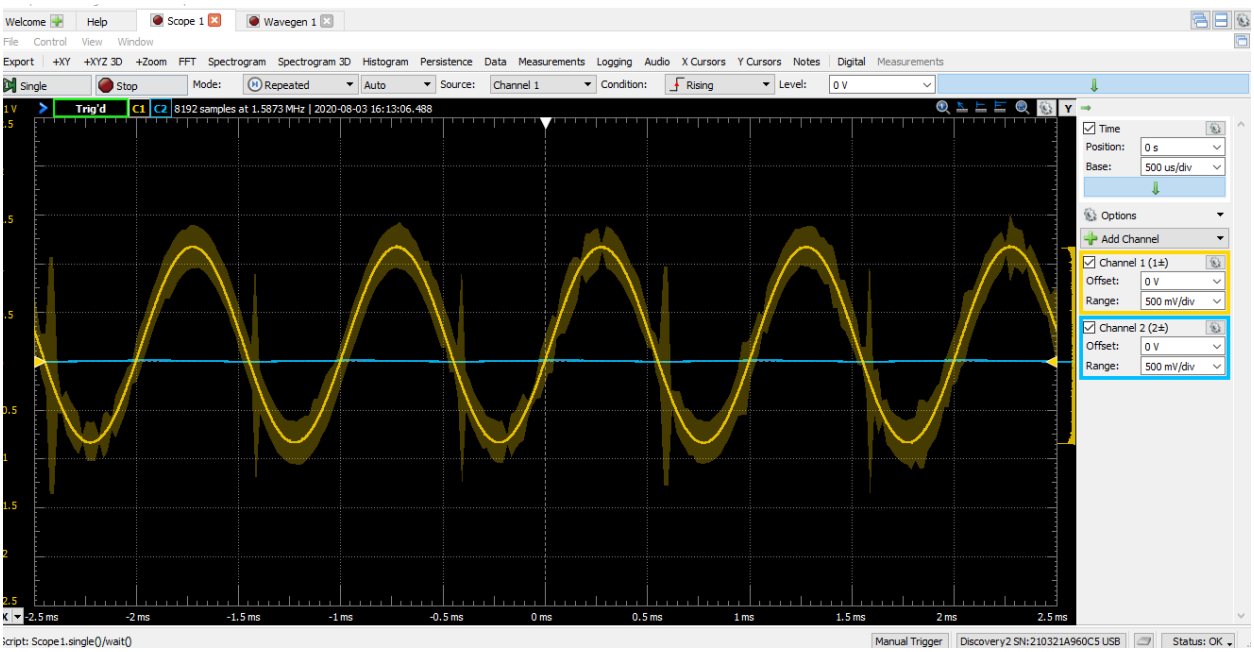
We then open the Logic tab from waveform. We added a channel, of DIO 0 then we ran it.



E) Arbitrary Wave Generator (AWG)



We connected the analog discovery, to the ground. We also connected the channel 1 to wave 1. We then ran both the wavegen 1 and scope together. To get



Conclusion:

This lab was not that bad toward the end, if you have everything working at the beginning. I have quite the issue with modifying the pin layout and everything since when I first started it was not giving me the default setting, so my pin layout was always blank. It too me a while before I got it to where I want it to be. Part 2 of the lab was the more interesting part since it was only adding on to the code to make it work the way we like it to. Overall, this lab is not bad as long as you don't get technical error in the setting up process.