Nathan Bunnell
Embedded Controller Programming with C
ECE-40291
Student ID: U08895857

Date: 5/17/2021

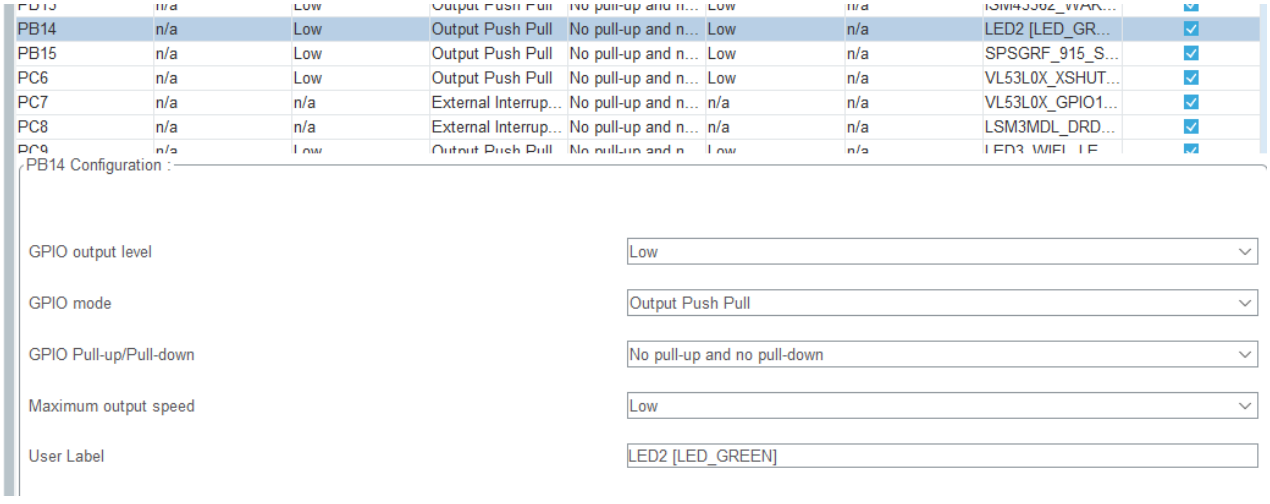
Assignment 5: GPIO

The following will document completion of the fifth assignment for ECE-40291, with the stated goals of:

1. Use STM32CubeIDE to generate the default code for the STM32 Discovery Board.
2. Edit, build, run, and debug the code that uses the HAL_GPIO API to toggle the LED2 On/Off when the blue button is pressed. The on/off toggle should happen in an interrupt service routine that runs when the blue button pressed.
3. The STM32 IOT Discovery board has an Arduino Connector. On that connector, connect pin D8 to the +(positive) side of an LED (the longer LED pin), then connect the -(negative) of the LED (the short LED pin) to a small (say 330 ohm) resistor, and then connect the resistor to the Arduino GND pin. Write code to blink the LED connected to D8 at a 1 second rate.

1. Use STM32CubeIDE to generate the default code for the STM32 Discovery Board

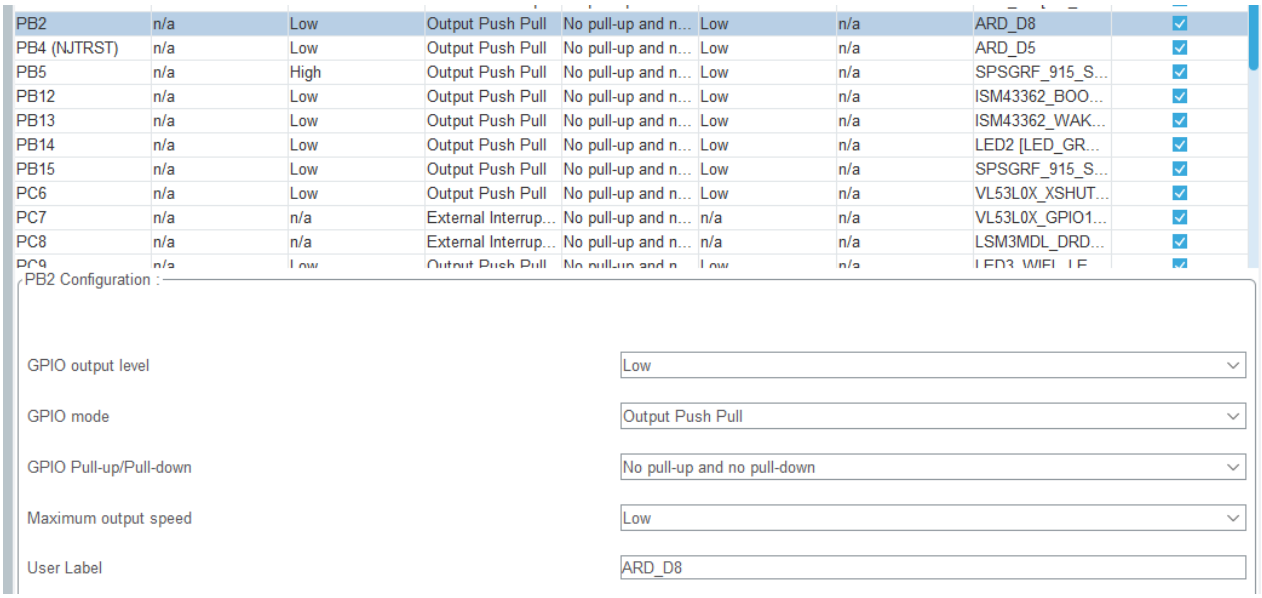
To begin, open the IDE and create a new project as per the methods documented in previous assignments. As we will be using the HAL in this example, there are no extra software packages to enable under the project configurator beyond the defaults, but we will want to confirm our IO config is what we expect for good measure. We can open the GPIO config and check LED2 that will be toggled by the blue pushbutton:



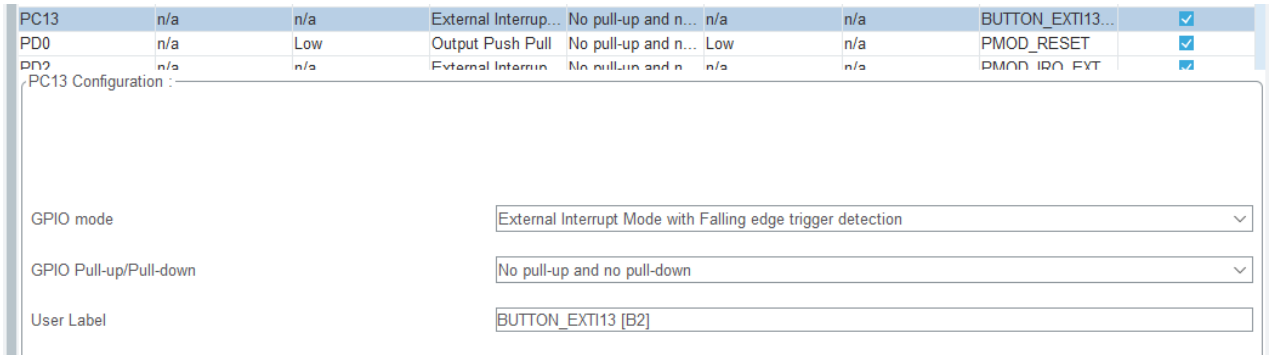
Nathan Bunnell
Embedded Controller Programming with C
ECE-40291
Student ID: U08895857

Date: 5/17/2021

As well as D8 on the Arduino connector, which will be tied to our blinking LED:



And finally, the blue pushbutton, which connects to EXTI13:



2. **Edit, build, run, and debug the code that uses the HAL_GPIO API to toggle the LED2 On/Off when the blue button is pressed. The on/off toggle should happen in an interrupt service routine that runs when the blue button pressed.**

With the configuration verified and default project code built, the on/off toggle is trivial to implement with the skills we've previously developed in this course. We will define a callback routine associated with the GPIO EXTI interface that is mapped to the specific pin, in this case PC13. Within the project's main.c Private user code section, we place the below function definition:

Nathan Bunnell
Embedded Controller Programming with C
ECE-40291
Student ID: U08895857

Date: 5/17/2021

```
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if (GPIO_Pin == GPIO_PIN_13)
    {
        HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
    }
}


/* USER CODE END 0 */
```

3. **The STM32 IOT Discovery board has an Arduino Connector. On that connector, connect pin D8 to the +(positive) side of an LED (the longer LED pin), then connect the - (negative) of the LED (the short LED pin) to a small (say 330 ohm) resistor, and then connect the resistor to the Arduino GND pin. Write code to blink the LED connected to D8 at a 1 second rate.**

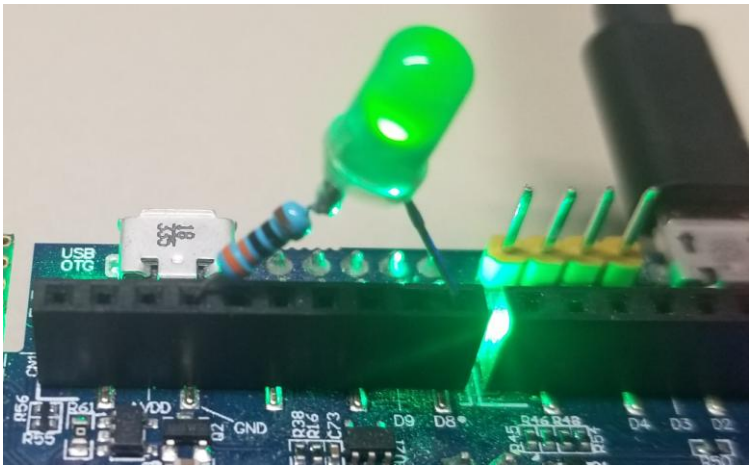
From here, the blink feature is even easier to implement, and can be accomplished as in previous assignments by calling the *HAL_Delay()* and *HAL_GPIO_TogglePin()* functions on the appropriate port and pin, which would bew PB2 or ARD_D8 if using the macros as seen below:

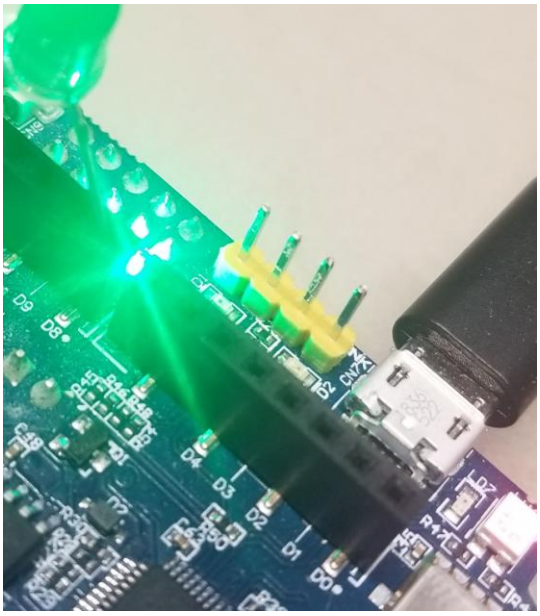
```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */

      HAL_GPIO_TogglePin(ARD_D8_GPIO_Port, ARD_D8_Pin);
      HAL_Delay(1000);
  }
  /* USER CODE END 3 */
}
```

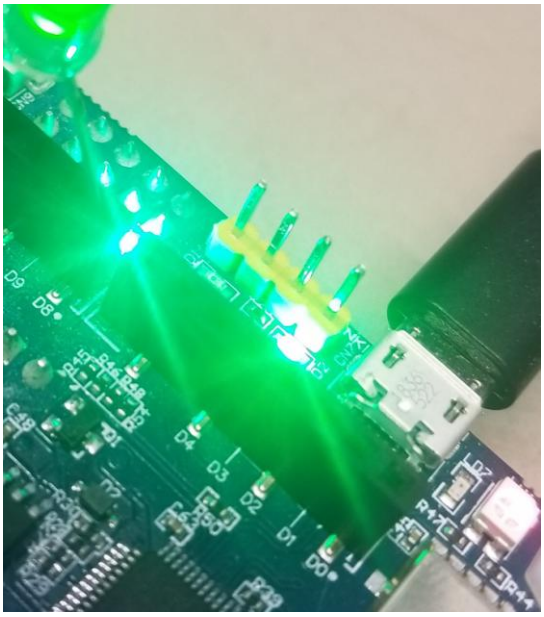
With the code succesfully compiled and downloaded, we can observer that the LED connected to D8 will toggle on and off at the expected rate, and that the LED2 will also toggle when the blue push button is pressed.

Nathan Bunnell
Embedded Controller Programming with C
ECE-40291
Student ID: U08895857

Date: 5/17/2021







Closing Thoughts

This assignment was straightforward to complete, especially so considering that the project default configuration set up the necessary GPIO to the functional modes we were targeting. In hindsight, I should have used a different colored LED to connect to the Arduino headers as there ended up being a lot of green on the board and it makes the images a bit confusing on what is the actual light source. Otherwise, it was extremely simple to reimplement code previously developed to accomplish the assignment goals.