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

Date: 5/8/2021

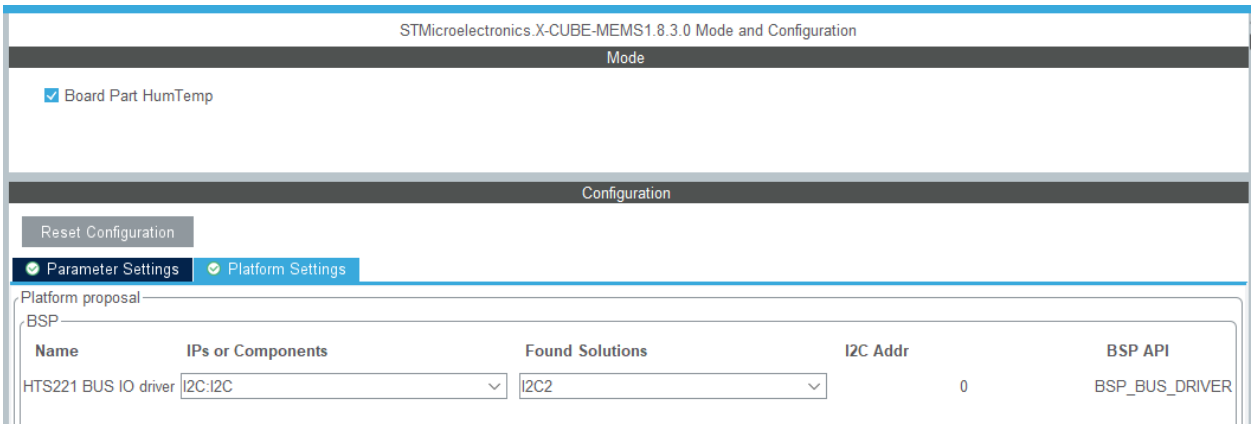
Assignment 4: BSP

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

1. Use STM32CubeIDE to generate the default code for the STM32 Discovery Board.
2. Include BSP code in your project
3. Use STM32CubeIDE to edit, build, run, and debug the code that uses the BSP API to (1) read the temperature, BSP_TSENSOR_ReadTemp(), read the blue push button status, BSP_PB_GetState(), and turn the LED on if button pressed, BSP_LED_On(), or turn LED off, BSP_LED_Off(), if button not pressed. Repeat this loop every 1 second.

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 the scope of this project includes taking readings from the on-board temperature sensor over one of the I2C busses, we will also need to add that configuration option through t he project configurator interface, selecting the appropriate hardware and device driver options as seen below.

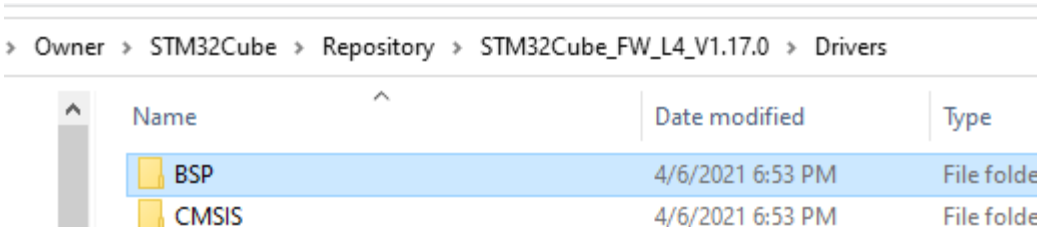


Nathan Bunnell
Embedded Controller Programming with C
ECE-40291
Student ID: U08895857

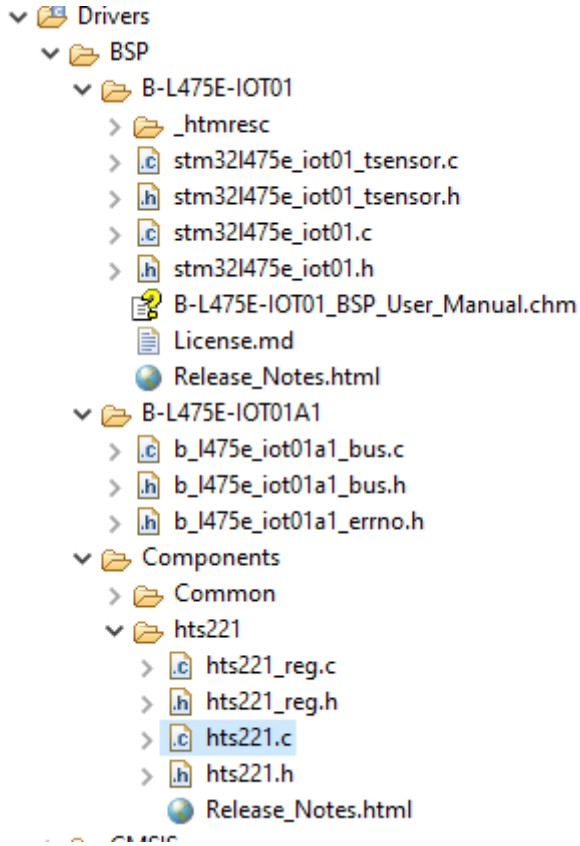
Date: 5/8/2021

2. Include BSP code in your project

To include the BSP in our project, we'll need to manually import the appropriate files from their location within the support directories installed with the IDE. In the case of my install, they were located at the below location with reference to my user account's ("Owner") home directory.



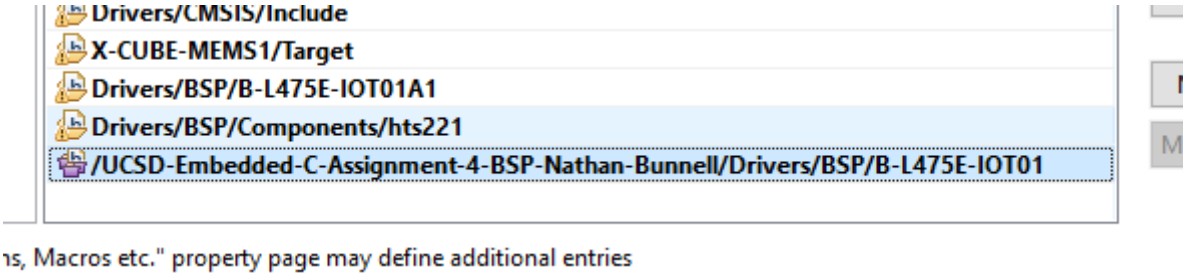
From here, we will manually copy into the project structure the appropriate files needed to read our temp sensor and utilize the BSP-level GPIO functions. Once imported, the project's Drivers folder should look something like this:



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

Date: 5/8/2021

At this point, we will need to add these BSP files to our project's path definitions so the linker will know where to look during compile time, as seen below.



We will also need to add the header files to our #include section, in addition to stdio.h and string.h, both of which will be required for the UART interface later.

```
23 /* Private includes ----------------------------
24 /* USER CODE BEGIN Includes */
25
26 #include "stm32l475e_iot01.h"
27 #include "stm32l475e_iot01_tsensor.h"
28
29 #include <stdio.h>
30 #include <string.h>
```

3. **Use STM32CubeIDE to edit, build, run, and debug the code that uses the BSP API to (1) read the temperature, BSP_TSENSOR_ReadTemp(), read the blue push button status, BSP_PB_GetState(), and turn the LED on if button pressed, BSP_LED_On(), or turn LED off, BSP_LED_Off(), if button not pressed. Repeat this loop every 1 second.**

At this point, we can develop what turns out to be fairly simple code to implement, thanks to the multi-layer abstraction provided by the BSP. The full project files will be attached with this submission but the summary is straightforward: we will call the appropriate BSP init functions for our sensor, *BSP_TSENSOR_Init()*, and our push button, *BSP_PB_Init(BUTTON_USER, BUTTON_MODE_GPIO)*. Then in our while(1) loop, read the temp sensor with calls to *BSP_TSENSOR_ReadTemp()*, format and print it out over the UART1 interface to the console, and finally check the state of the blue user button with *BSP_PB_GetState(BUTTON_USER)*. If it is pressed, turn on LED2 with *BSP_LED_On(LED2)*, else, leave it off, then restart the loop.

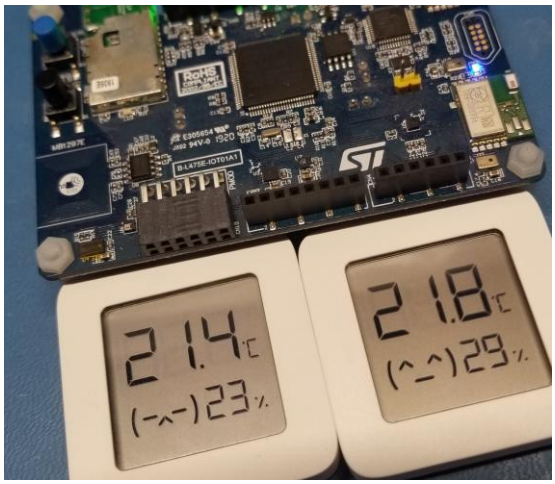
Nathan Bunnell
Embedded Controller Programming with C
ECE-40291
Student ID: U08895857

Date: 5/8/2021

With this project compiled without errors, we can flash it to our dev board and connect via serial console to observe functionality. Measuring the ambient temp, we see feedback of 22 deg C, which sounds about right compared to how the room *feels*.

```
Loop #1. HST221 reading 22 degrees C
Loop #2. HST221 reading 22 degrees C
Loop #3. HST221 reading 22 degrees C
Loop #4. HST221 reading 22 degrees C
Loop #5. HST221 reading 22 degrees C
```

I had wanted to compare these readings to the thermocouple probe of my DMM but couldn't find it in my equipment box. I do have a handful of very cheap import temp/humidity sensors laying around from another development project that were reading a similar value so while I wouldn't call the a calibrated reference point, the similarity between all three falls under what I would call close enough for this application.



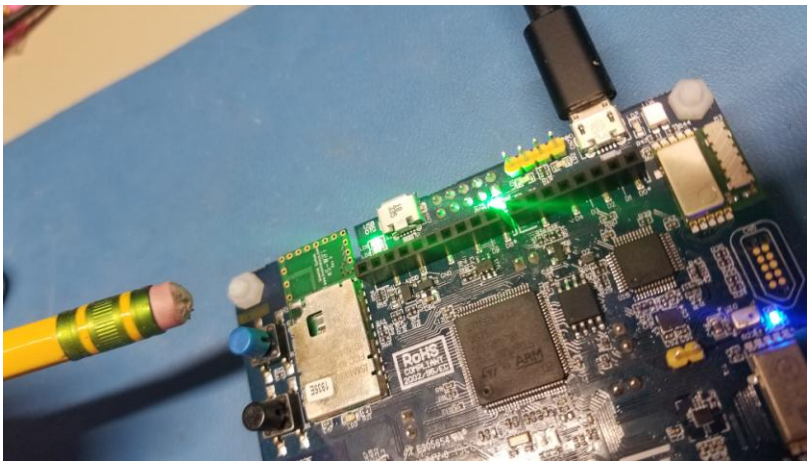
As another test, I held my a space heater over the top of the board for about 30 seconds and was abale to observe a steady increase in temp feedback from the board. Again, no real controlled reference point in this case but it tracks to within reasonable expectations.

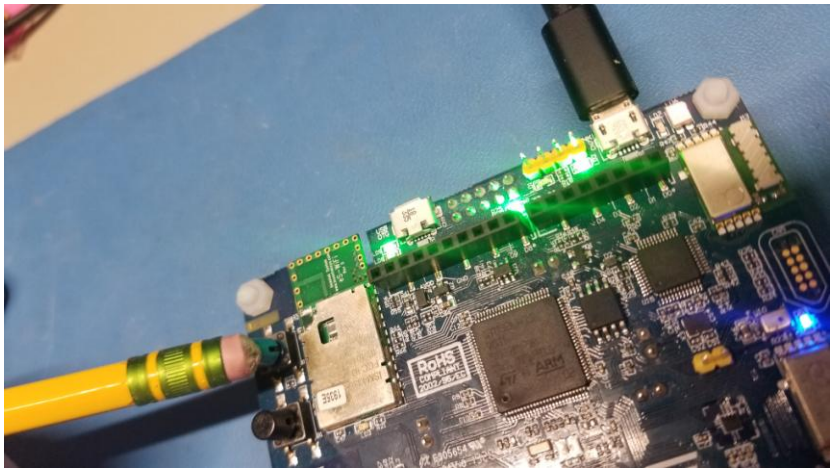
```
Loop #86. HST221 reading 37 degrees C
Loop #87. HST221 reading 38 degrees C
Loop #88. HST221 reading 38 degrees C
Loop #89. HST221 reading 39 degrees C
Loop #90. HST221 reading 40 degrees C
Loop #91. HST221 reading 41 degrees C
Loop #92. HST221 reading 42 degrees C
Loop #93. HST221 reading 42 degrees C
```

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

Date: 5/8/2021

For the final part of the assignment, simply noting that LED2 is off when the blue button is not pressed versus on when it is pressed serves as sufficient proof that that portion of the code functions to design.





Closing Thoughts

Much like the LL APIs previously studied, the BSP was a level of abstraction that I was not previously familiar with. For the purposes of this assignment, it was extremely simple to use, with the entire development process being much shorter than assignments using only the HAL or LL APIs. One hurdle I discovered, which might be a version issue or (probably) user error, was that that portion of the BSP files that were supposed to add themselves to the project automatically pulled in filed from a different directory, the dir “B-L475E-IOT01A1” seen in the reference screenshot previously. Overcoming, this was as simple as following the manual import process that would need to be completed for the other support files so it was more of an unexpected speed bump than any kind of real issue.