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

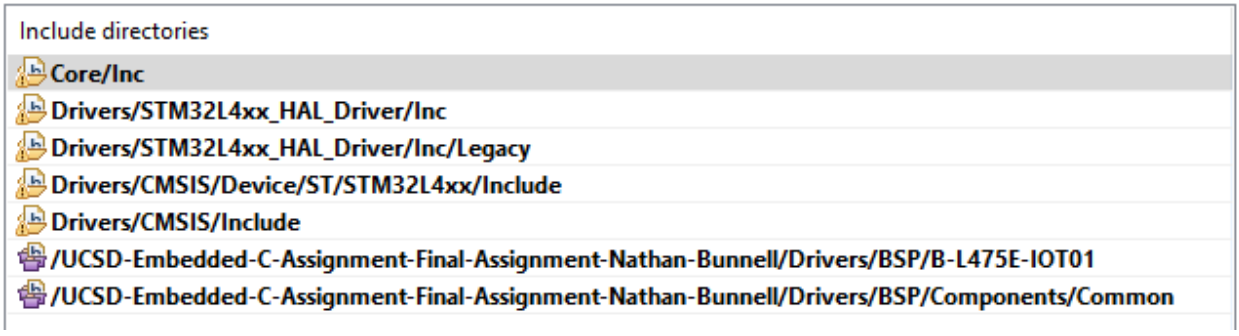
Date: 6/3/2021

Course Final Project

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

1. LL APIs: (a) get flash size LL_GetFlashSize(); (b) get the device unique ID, LL_GetUID_Wordn(); 3) toggle the LED, LL_GPIO_TogglePin() at a 1 second rate. Display the Flash Size and GUID only when demo begins, but keep flashing the LED every 1 second until the Blue Button is pressed to advance to next demo
2. HAL APIs: (a) get the device ID, HAL_GetDEVID(), (b) read the device unique ID, HAL_GetUIDwn(), and (c)toggle the LED, HAL_GPIO_TogglePin() at a 2 second rate, using HAL_Delay() to sleep for the 2 seconds. Display the Dev ID info only when demo 2 begins, but keep flashing the LED every 2 second until the Blue Button is pressed to advance to next demo.
3. BSP APIs: (a) read the temperature, BSP_TSENSOR_ReadTemp() (b) turn the LED on every 3 seconds with BSP_LED_On() (c) turn LED off, every 3 seconds with BSP_LED_Off(). In other words the LED should blink on/off at a 3-second rate (3 seconds on, 3 seconds off). Continue this until the Blue Button pressed to advance to the next demo.
4. Bonus Demo: demonstrate an optional I/O device that is on the STM board. Or, connect additional I/O devices to the Arduino connectors

Prior to deploying any of the demos, we will build our default project, enable UART4 to our standard 115.2, 8, N, 1 config, and then follow the process worked out in previous assignments to add the BSP packages to our project structure. We'll also need to pull in the LL header files in our #includes section to allow use of both the default HAL config as well as the LL calls later.



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

Date: 6/3/2021

```
24  /* USER CODE BEGIN Includes */
25
26  #include "stm32l4xx_ll_utils.h"
27  #include "stm32l4xx_ll_gpio.h"
28
29  #include "stm32l475e_iot01.h"
30  #include "stm32l475e_iot01_tsensor.h"
31
32  #include <stdio.h>
33  #include <string.h>
```

Next, we can build the infrastructure to call and move between demos. We'll manage that via a callback function tied to the blue user button as an external interrupt. When the button is pressed, the *demoCount* variable will be incremented, rolling around from 4 to 1. We'll also print a message to the console with the value of the counter.

```
66  /* USER CODE BEGIN PV */
67
68  // Global counter variable to track current demo #
69  uint8_t demoCount = 1;
```

```
 94  /* USER CODE BEGIN 0 */
 95
 96  void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
 97  {
 98      if (GPIO_Pin == GPIO_PIN_13)
 99      {
100          if (demoCount >= 4)
101          {
102              demoCount = 1;
103          }
104          else
105          {
106              demoCount += 1;
107          }
108
109          char buffer[100] = {0};
110          snprintf(buffer, sizeof(buffer), "\nMoving to demo #%d\n", demoCount);
111          HAL_UART_Transmit(&huart4, (uint8_t*) buffer, strlen(buffer), 1000);
112      }
113  }
```

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

Date: 6/3/2021

The final piece of infrastructure will be within the main *while(1)* loop, where we pass the value of *demoCount* into a simple switch structure and then print out the associated demo title before calling it.

```
278     while (1)
279     {
280     /* USER CODE END WHILE */
281     /* USER CODE BEGIN 3 */
282         // Define strings to structure prompt around
283         char* cliResponse = "\0";
284 
285         // Evaluate input
286         switch (demoCount)
287         {
288             case 1:
289                 cliResponse = "\nLL demo:\n";
290                 HAL_UART_Transmit(&huart4, (uint8_t*) cliResponse, strlen(cliResponse), 1000);
291                 do_LL_demo();
292                 break;
293 
294             case 2:
295                 cliResponse = "\nHAL demo:\n";
296                 HAL_UART_Transmit(&huart4, (uint8_t*) cliResponse, strlen(cliResponse), 1000);
297                 do_HAL_demo();
298                 break;
299 
300             case 3:
301                 cliResponse = "\nBSP demo:\n";
302                 HAL_UART_Transmit(&huart4, (uint8_t*) cliResponse, strlen(cliResponse), 1000);
303                 do_BSP_demo();
304                 break;
305 
306             case 4:
307                 cliResponse = "\nBonus demo:\n";
308                 HAL_UART_Transmit(&huart4, (uint8_t*) cliResponse, strlen(cliResponse), 1000);
309                 do_bonus_demo();
310                 break;
311 
312             default:
313                 HAL_UART_Transmit(&huart4, (uint8_t*) cliResponse, strlen(cliResponse), 1000);
314                 break;
315         }
316     }
317   /* USER CODE END 3 */
```

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

Date: 6/3/2021

1. LL API Demo

On boot, we'll default to the LL demo. In this mode, we leverage those APIs to get and print the device's flash size, unique ID number, and then jump into a loop where we blink LED2 at a one second rate until the user button is pressed.

```
115  /*
116   * (a) get flash size  LL_GetFlashSize()
117   * (b) get the device unique ID, LL_GetUID_Wordn()
118   * (c) toggle the LED, LL_GPIO_TogglePin() at a 1 second rate
119   *
120   * Display the Flash Size and UID only when demo begins,
121   * but keep flashing the LED every 1 second until the Blue Button is
122   * pressed to advance to next demo.
123   *
124   */
125  void do_LL_demo(void)
126  {
127      char buffer[100] = {0};
128
129      uint32_t flashSize = LL_GetFlashSize();
130
131      snprintf(buffer, sizeof(buffer), "\tFlash size: %lu\n", flashSize);
132      HAL_UART_Transmit(&huart4, (uint8_t*) buffer, strlen(buffer), 1000);
133
134      uint32_t uidWord[3];
135      uidWord[0] = LL_GetUID_Word0();
136      uidWord[1] = LL_GetUID_Word1();
137      uidWord[2] = LL_GetUID_Word2();
138
139      snprintf(buffer, sizeof(buffer), "\tUID: %lu%lu%lu\n", uidWord[0], uidWord[1], uidWord[2]);
140      HAL_UART_Transmit(&huart4, (uint8_t*) buffer, strlen(buffer), 1000);
141
142      while(demoCount == 1)
143      {
144          LL_GPIO_SetOutputPin(GPIOB, GPIO_PIN_14);
145          HAL_Delay(1000);
146          LL_GPIO_ResetOutputPin(GPIOB, GPIO_PIN_14);
147          HAL_Delay(1000);
148      }
149  }
```

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

Date: 6/3/2021

2. HAL API Demo

The second demo will get the device ID value and then call for the unique ID number again, printing them then jumping into another blinky loop at a two second rate until the user button is pressed again, this time with the HAL APIs.

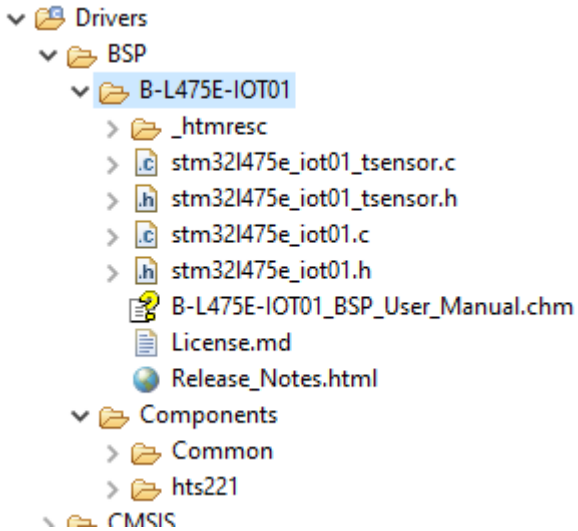
```
151  /*
152   * (a) get the device ID, HAL_GetDEVID()
153   * (b) read the device unique ID, HAL_GetUIDwn()
154   * (c)toggle the LED, HAL_GPIO_TogglePin() at a 2 second rate, using HAL_Delay() to sleep for the 2 seconds.
155   *
156   * Display the Dev ID info only when demo 2 begins, but keep flashing
157   * the LED every 2 second until the Blue Button is pressed to
158   * advance to next demo.
159   *
160   */
161  void do_HAL_demo(void)
162  {
163      char buffer[100] = {0};
164
165      uint32_t devID = HAL_GetDEVID();
166
167      snprintf(buffer, sizeof(buffer), "\tDevice ID: %lu\n", devID);
168      HAL_UART_Transmit(&huart4, (uint8_t*) buffer, strlen(buffer), 1000);
169
170      uint32_t uidWord[3];
171      uidWord[0] = HAL_GetUIDw0();
172      uidWord[1] = HAL_GetUIDw1();
173      uidWord[2] = HAL_GetUIDw2();
174
175      snprintf(buffer, sizeof(buffer), "\tUID: %lu%lu%lu\n", uidWord[0], uidWord[1], uidWord[2]);
176      HAL_UART_Transmit(&huart4, (uint8_t*) buffer, strlen(buffer), 1000);
177
178      while(demoCount == 2)
179      {
180          HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
181          HAL_Delay(2000);
182      }
183  }
```

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

Date: 6/3/2021

3. BSP API Demo

The third demo will leverage the BSP interface, so assuming that it was added at the beginning of the project, we can trim out the unnecessary files to and ensure that we do call the appropriate init() function prior to reading from the sensor.




```
270
271   // Init the temp sensor
272   BSP_TSENSOR_Init();
273
```

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

Date: 6/3/2021

The do_BSP_demo is structured similarly to the previous two, in this case calling the *BSP_TSENSOR_ReadTemp()* function, trimming the value to be an integer, and then entering another blink loop, this time at a three second rate.

```
193  void do_BSP_demo(void)
194  {
195      // Read the temperature sensor, trim the mantissa to give an integer value
196      float temperature = BSP_TSENSOR_ReadTemp();
197      int value = temperature;
198
199      char buffer[100];
200      snprintf(buffer, sizeof(buffer), "\tHTS221 reading %d degrees C\n", value);
201      HAL_UART_Transmit(&huart4, (uint8_t *)buffer, strlen(buffer), 1000);
202
203      while(demoCount == 3)
204      {
205          BSP_LED_On(LED2);
206          HAL_Delay(3000);
207          BSP_LED_Off(LED2);
208          HAL_Delay(3000);
209      }
210  }
```

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

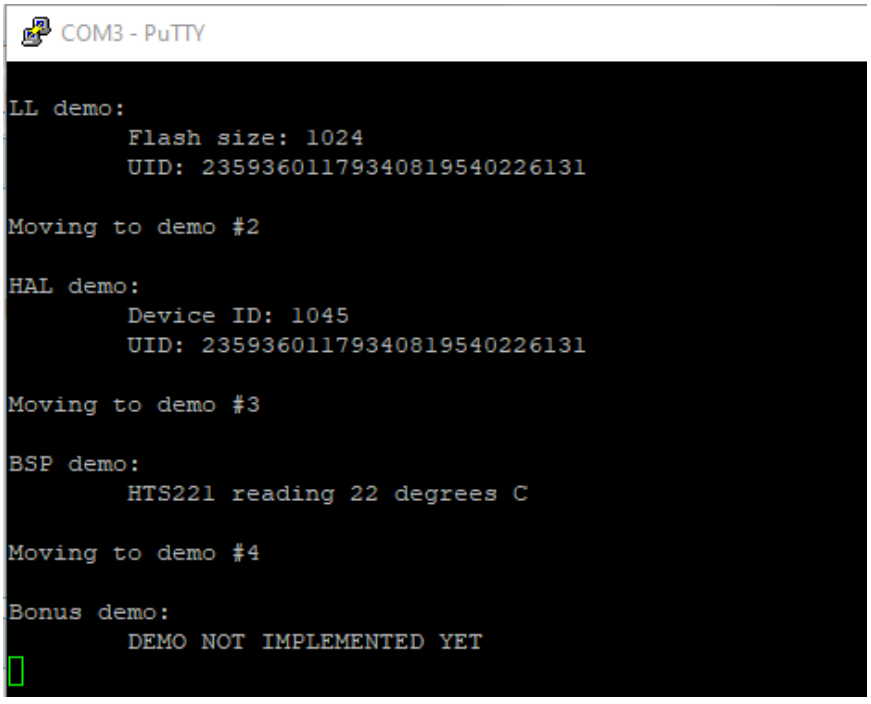
Date: 6/3/2021

4. Bonus Demo

I had intended to include reading an RTC module as a bonus demo, and built the interface for it, but didn't find time to develop the code for the I2C side. I also intended to use this module as a primary feature in the final project for ECE-40293 and might come back to it in this project once I get it working. Until then, this placeholder code simply prints out that the demo hasn't yet been implemented.

```
213  /*
214   * (4) connect an I2C or SPI device to the Arduino pins, and read the info from those devices
215   */
216  void do_bonus_demo(void)
217  {
218      char* cliResponse = "\tDEMO NOT IMPLEMENTED YET\n";
219      HAL_UART_Transmit(&huart4, (uint8_t*) cliResponse, strlen(cliResponse), 1000);
220
221      while(demoCount == 4)
222      {
223          HAL_Delay(1000);
224      }
225  }
226
```

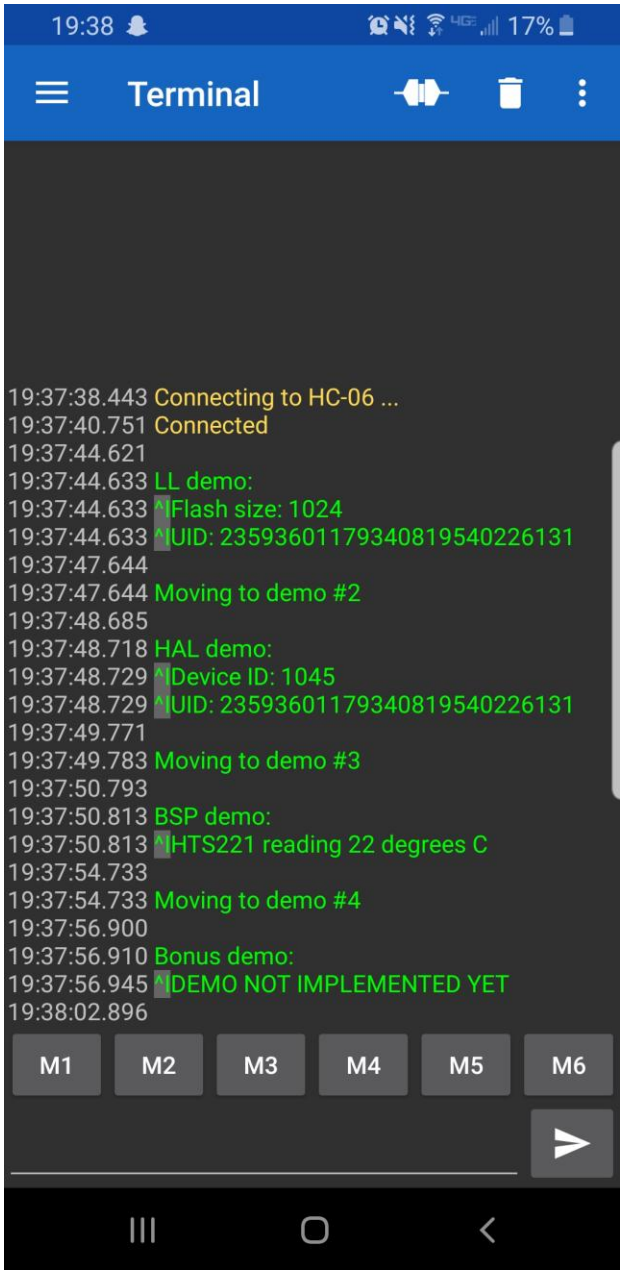
With our demos defined, we can compile and flash to our board and then connect to the serial port. Cycling through with the blue button, we can observe LED2 changing its frequency as we also see each demo print out its results.



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

Date: 6/3/2021

And I also hadn't planned on including this feature under this project, but I decided that since the code was already built, I would attempt to prove the concept. Disconnecting my USB-serial UART adapter and replacing it with an HC-06 based Bluetooth-serial adapter, I re-flashed the project with UART4 configured to 9600 baud and was able to monitor the same demos from a terminal app on my phone, with minor formatting issues. This is also intended to be deployed in the final project for ECE-40293, but again, this was a quick way to test if the module was still functional after sitting in my parts bin for probably six or seven years.



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

Date: 6/3/2021

Closing Thoughts

This course has been a fantastic learning experience for me. I was initially leery of using a vendor supplied IDE but after having spent the past nine weeks working with it, I can appreciate the value it would offer to a professional developer, especially in areas like the BSP or HAL interfaces to the various peripherals. I still scratch my head at some of the behaviors exhibited by the IDE but maybe that's part of using a new tool. Additionally, being able to spend time constantly developing and deploying code for this piece of hardware has just been fun. As I've discussed before, I spend all day with either ladder logic or scripting languages. While all of my PLCs are tied directly to real world I/O, everything about working with them is so abstracted and virtualized away from the processor-level that it was a refreshing change of pace to interact with the real hardware on the Discovery board. Finally, the constant activity on the Discussion Board was a highlight; it has been great to be able to share experiences and bounce ideas off of the other students taking this course.