(CSE211s) INTRO. TO EMBEDDED SYSTEMS

PROJECT DOCUMENTATION

Team members:

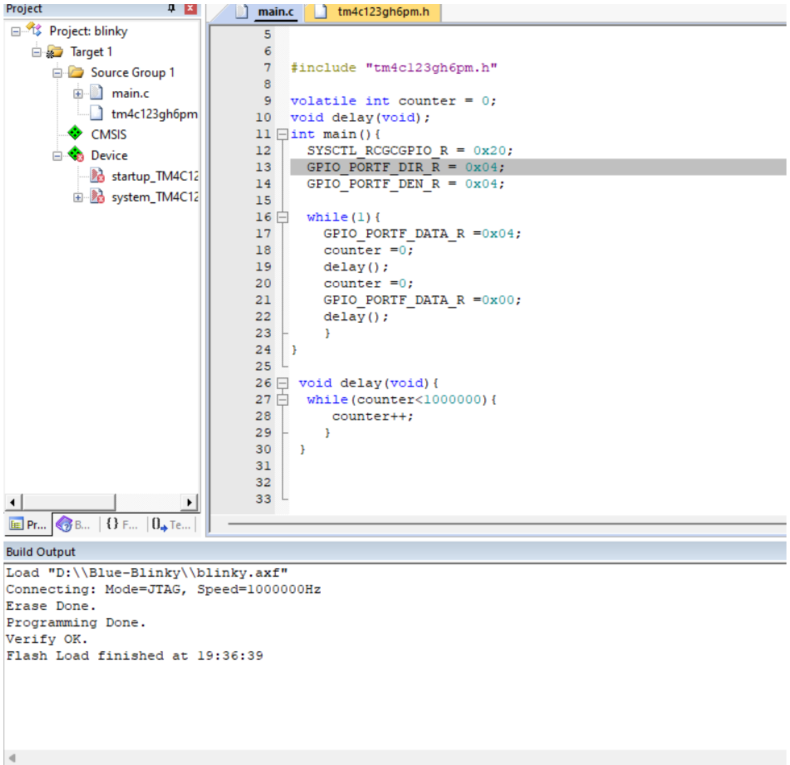
- Ahmed Khaled Mohamed Abdulrahman Sallam 2100328
- Abdulrahman Ahmed Saeed Abdelmaged 2100811
- Abdulrahman Ezz 2101000
- Omar Ashraf Abdulsatar 2100354
- Shorouk Amr Aly Mustafa 2100539
- Khaled Alaa El-Din El-Sayed Doma 2101422
- Mohamed Atta El-Sayed Atta 2101521
- Kareem Mostafa Hamed El-Hanafy 2101097
- Ahmed Samir Helmy 2101458

1. Project description:

In our current endeavor, we're tasked with creating a GPS tracking system. This system is crafted using the principles of embedded C programming. It operates by acquiring real-time location data from a TM4C123G LaunchPad microcontroller that's on the move, starting from the moment it's powered on until it arrives at its final destination. The gathered information is then seamlessly relayed to a personal computer, where it's displayed on a mapping application for easy visualization.

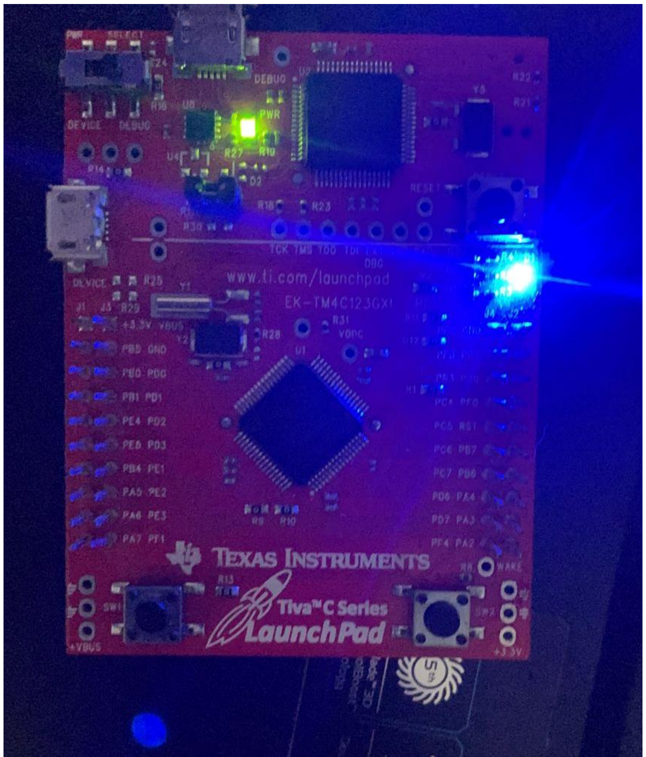
2. Project in Action (Screenshots):

This screenshot was captured real-time, as we tested our board and a starter code for the LED:



```c


#include "tm4c123gh6pm.h"

volatile int counter = 0;
void delay(void);
int main(){
  SYSCTL_RCGCGPIO_R = 0x20;
  GPIO_PORTF_DIR_R = 0x04;
  GPIO_PORTF_DEN_R = 0x04;

  while(1){
    GPIO_PORTF_DATA_R =0x04;
    counter =0;
    delay();
    counter =0;
    GPIO_PORTF_DATA_R =0x00;
    delay();
    }
}

 void delay(void){
  while(counter<1000000){
     counter++;
    }
 }



```

```
Load "D:\\Blue-Blinky\\blinky.axf"
Connecting: Mode=JTAG, Speed=1000000Hz
Erase Done.
Programming Done.
Verify OK.
Flash Load finished at 19:36:39
```



3. Source Code:

Entire project source code can be found on our team’s GitHub repo: Omar-26/GPS_Tracking_System: Embedded Systems GPS Distance Tracking System (github.com)

The tree structure of our repo is explained briefly on this page:

- **Application Layer (APP)**: This is the layer where the primary sequence of the software resides. It is specific to the software in use.
- **Hardware Abstraction Layer (HAL)**: This layer offers a high-level interaction with the hardware. It enhances the portability of the application code, allowing the same code to function with various hardware by merely using a different HAL implementation.
- **Microcontroller Abstraction Layer (MCAL)**: This layer is responsible for managing the microcontroller hardware. It encompasses our primary drivers, such as GPIOs, communication interfaces (SPI, I2C, UART), ADCs, and so on.
- **Library (LIB)**: This includes third-party or proprietary libraries that the project may rely on. These libraries offer a range of functions and utilities that are not specific to the hardware or the application but are utilized by them. Examples include data structures, mathematical functions, or communication protocols.