

# (CSE211s) INTRO. TO EMBEDDED SYSTEMS

## PROJECT DOCUMENTATION

### Team members:

- Shehab Mahmoud Salah (2100320)
- Abdelrahman Hany Mohamed (2100627)
- Youssef Ahmed Mohamed (2101006)
- Omar Mamon Hamed (2100767)
- Seif El Din Tamer Shawky (2100268)
- Seif Eldeen Ahmed Abdulaziz (2100339)
- Habiba El-sayed Mowafy (2100792)
- Moaz Ragab (2100938)
- Mohamed Waleed Sayed (2100623)
- Ahmed Ashraf Ali (2100255)

### 1. Project Outline:

In the given project, we are assigned to a **GPS TRACKING SYSTEM** developed using embedded C programming, by gathering real-time positional coordinates while a microcontroller is in motion (using TM4C123G LaunchPad) after power-on until a destination point is reached. The collected data will be efficiently transferred to a personal computer and visualized on a map application.

### 2. Project in Action (Screenshots):

The following screenshots were captured real-time, as we tested our **GPS** module:

The screenshot displays the Keil IDE environment. The main window shows the source code for the GPS tracking system. The code includes initialization for the LED and UART, and a main loop that reads GPS data and calculates the distance to the destination. The Watch window shows the current state of the program variables.

**Code Snippets:**

```
// Open Red LED and stop when destination is reached
MCAL_GPIO_SetPinValue(GPIO_PORTF, GPIO_PIN1, GPIO_HIGH);
while(1);

// Initializing Flashing LED and UART for GPS
RED_led_Init();
UART2_Init();

// Poll until a valid first GPS reading is found
while((HAL_GPS_Read(&cfg, &first_latitude, &first_longitude)) == 0);
current_latitude = first_latitude;
current_longitude = first_longitude;

while(1){
    // Get the distance every GPS tick
    distance = computeDistance(current_latitude, current_longitude, first_latitude, first_longitude);
    while((HAL_GPS_Read(&cfg, &current_latitude, &current_longitude)) == 0);
```

**Watch 1 (Left):**

| Name              | Value                   | Type       |
|-------------------|-------------------------|------------|
| Received_Command  | <cannot evaluate>       | uchar      |
| readData          | <cannot evaluate>       | uchar      |
| distance          | 0                       | float      |
| current_latitude  | 0                       | float      |
| current_longitude | 0                       | float      |
| first_latitude    | 0                       | float      |
| first_longitude   | 0                       | float      |
| GPS_Sentence      | 0x20000060 GPS_Sente... | uchar[100] |

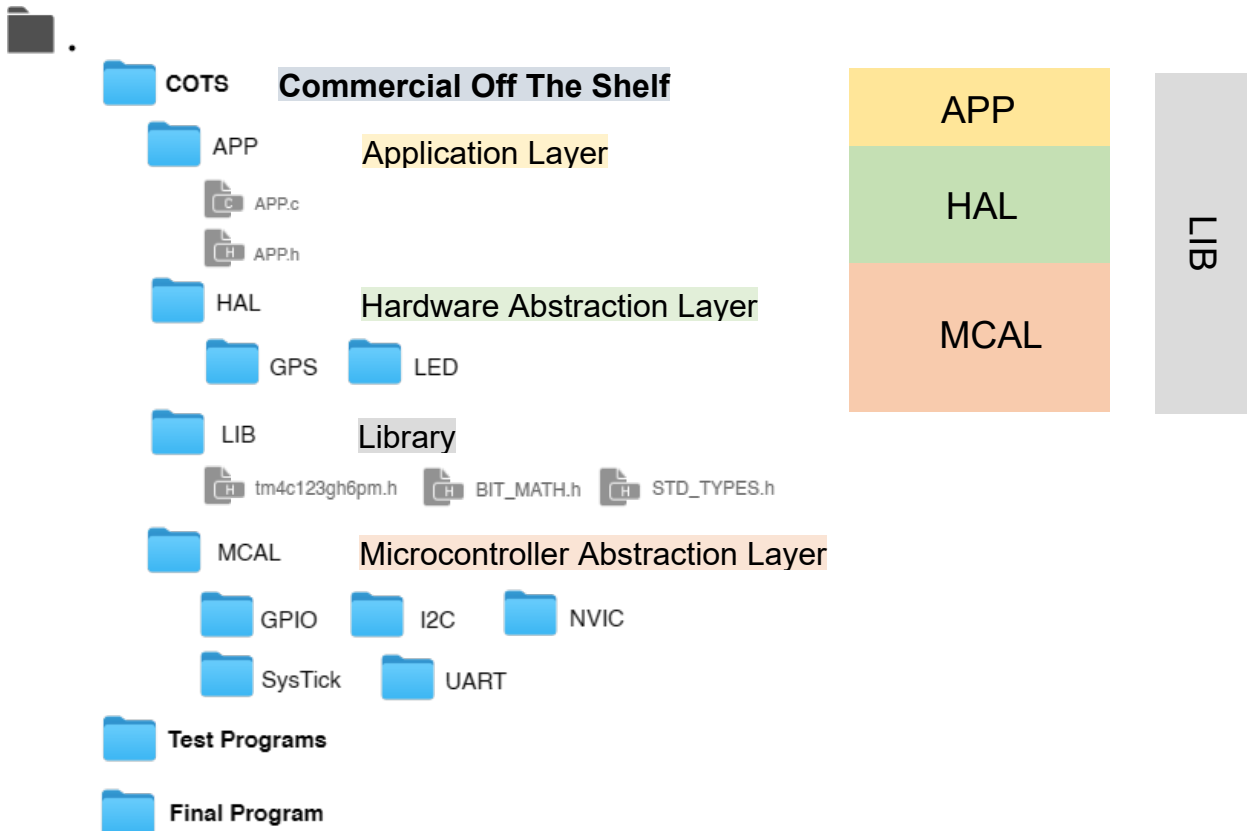
**Watch 1 (Right):**

| Name              | Value                   | Type       |
|-------------------|-------------------------|------------|
| Received_Command  | <cannot evaluate>       | uchar      |
| readData          | <cannot evaluate>       | uchar      |
| distance          | 0.657449961             | float      |
| current_latitude  | 3002.54297              | float      |
| current_longitude | 3123.6582               | float      |
| first_latitude    | 3002.54297              | float      |
| first_longitude   | 3123.65796              | float      |
| GPS_Sentence      | 0x20000060 GPS_Sente... | uchar[100] |

### 3. Source Code:

Entire project source code can be found on our team's GitHub repo:  
<https://github.com/MashaWaleed/GPS-System-TIVAC-CSE211>

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



The layers, as depicted in the figures above are demystified as follows:

#### (APP) Application Layer:

It's where the main flow of the program resides and is software specific.

#### (HAL) Hardware Abstraction Layer:

provides a high-level interface to the hardware. It makes the application code more portable as the same application code can work with different hardware just by using a different HAL implementation.

#### (MCAL) Microcontroller Abstraction Layer:

manages the microcontroller hardware. It includes our main drivers, i.e: GPIOs, communication interfaces (SPI, I2C, UART), ADCs, etc.

#### (LIB) Library:

Include third-party or proprietary libraries that the project might depend on. They provide various functions and utilities that are not specific to the hardware or the application but are used by them, such as data structures, math functions, or communication protocols.