GLOBAL POSITIONING SYSTEM (GPS)

CSE 211s Introduction to Embedded Systems

Spring 2025

- Ain Shams University
- Faculty of Engineering

Contents

TEAM MEMBERS	ID	Contribution
Nour Ahmed Khalaf Ahmed	2200715	UART, LED_BUZZER, LCD connection & Test, Report
Rana Gamal Reda	2200749	GPIO, SWITCH_BTN, GPS test, Report
Sherif ahmed Abdelfattah	2200176	LCD, SYSTICK, Report, Packaging
Karim Hosam Ahmed Ali	2201405	GPS, APP, Managed GitHub repository, TIVA test
Ahmed Emad Mohamed Zaghloul	2201374	GPS, APP, TIVA test, GPS test
Ziad Amr Fathy	2200081	TIMER, bit_utilies, TIVA test, SWITCHS Connection
Mariam Tarek Samir	2200167	GPIO, LCD, Led_buzzer connection, Report

LINKS:

- **GitHub repositories:**

https://github.com/Karim-Hosam/Smart_GPS_System

- **Video:**

https://drive.google.com/drive/folders/1--VJLWhQEizr60iRE9g88UFimhKUpodO

Introduction

In today's location-aware world, GPS technology has become fundamental for navigation and tracking applications. This project implements a real-time landmark proximity detection system using the TM4C123G LaunchPad microcontroller and a GPS receiver module. Designed specifically for our faculty campus, the system identifies nearby academic buildings and displays them on an LCD as the user moves between locations.

The core functionality revolves around acquiring GPS coordinates, comparing them against pre-stored landmark locations (including at least 5 faculty halls), and determining the nearest one using distance calculations. The system provides immediate visual feedback through an LCD display and optional audible/visual alerts using a buzzer and LED. For portability, it operates autonomously on battery power.

This implementation serves as an excellent case study in embedded C programming**,** demonstrating:

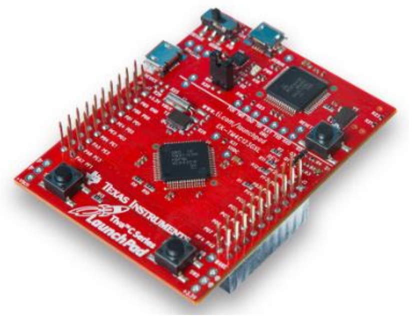
UART communication with GPS modules for NMEA sentence parsing

GPIO configuration for user inputs (buttons) and outputs (LCD, LEDs)

Real-time processing of location data with periodic updates

Low-power considerations for battery operation

Hardware Components



1. TM4C123G Launchpad (Microcontroller)
2. GPS Module
3. Personal Computer z
4. Connecting Cables (USB, Serial)
5. LCD
6. Battery & Power Bank (Dual power source)
7. LED
8. BUZZER
9. Bread Board
10. Push buttons

Software Tools

1. Keil IDE (C Development Environment)
2. TM4C123G Header File

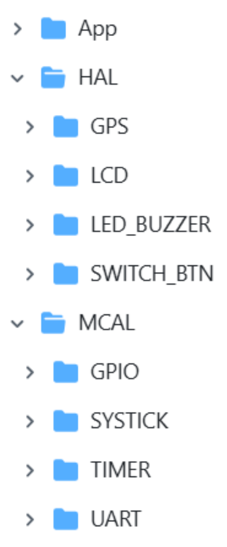
Project Architecture

- **Layered Design**

APP (Application Layer)
Main logic for state machine, landmark matching, user input handling

MCAL (Microcontroller Abstraction Layer)
Direct register-level access to peripherals (GPIO, UART, Timer, SYSTICK).

HAL (Hardware Abstraction Layer)
Drivers for LCD, GPS parser, LED, buzzer, and SWITCH_PIN.



Code Structure & Key Modules

❖ APP:

APP.h

APP.h is the header file for the application layer.

- It includes all necessary header files from:
  - Standard C libraries (stdlib.h, stdio.h, string.h, stdint.h)
  - Microcontroller-specific utilities (TM4C123.h, bit_utilities.h)
  - MCAL layer modules (TIMER, GPIO, UART, SYSTICK)
  - HAL layer modules (GPS, LCD, LED_BUZZER, SWITCH_BTN)
- It defines several macros for reading specific button inputs and checking GPS data validity:
  - Save_LAT_LOG_BTN, SWITCH_STATE_BTN, SWITCH_PLACE_BTN, RESET_TIMER_BTN
  - DATA_RECIEVED_FROM_GPS_CORRECTLY: A condition macro to check if valid GPS coordinates were received.

APP.C

This is the source file that contains the core logic of the system. It performs the following key tasks:

- Initialization of UART, LCD, GPIO ports, and system timers.
- GPS Integration: Receives and parses GPS coordinates using UART. The latitude and longitude are stored and converted from string to float.
- System States:

    - State 1: Displays the nearest predefined place and its distance.
    - State 2: Displays the distance to a manually saved location.
    - State 3: Allows the user to manually cycle through and select a place.
- LCD Output: Dynamically updates the LCD with current state information, place name, and distance.
- User Input Handling: Responds to button presses to change states, save locations, reset the timer, or switch manually between places.
- Alerts: Triggers alerts if the device gets close to a defined location using buzzer/LED via Alert_If_Close().

The structure of APP.c showcases how a complete embedded application can be managed in an organized state machine, integrating both real-time data (from GPS) and user input in an interactive system.

❖ MCAL:

GPIO

GPIO (General Purpose Input/Output) pins on the Tiva C microcontroller can be set as inputs (to read buttons or sensors) or outputs (to control LEDs, motors, etc.). They are controlled via software and are essential for connecting the microcontroller to external devices.

Functions used:

GPIO_InitPort	Initializes a GPIO port by enabling its clock, unlocking it, and configuring its digital settings. Steps: • Enable clock for the specified port. • Wait until the port is ready. • Unlock the port (if needed). • Enable access to selected pins. • Disable analog functions and alternate functions • Enable digital I/O for the selected pins.
GPIO_SetPinValue	Writes a value to a pin: 'S' = Set (logic 1/HIGH) ,'C' = Clear (logic 0/LOW).
GPIO_SetPinDirection	Sets the direction of a specific pin on a GPIO port as either input or output.
GPIO_SetPortValue	Writes a digital value (HIGH or LOW) to a specific GPIO pin, typically used to turn an LED on or off.
GPIO_GetPinValue	**Reads the digital input value (HIGH or LOW) from a specific GPIO pin. Useful for checking the state of buttons or sensors.**
Button_Read	A custom function that checks whether a specific button is pressed or not. It likely wraps around GPIO_GetPinValue with debouncing or mapping.

SYSTICK

The SYSTick timer is a built-in 24-bit down-counter in the Tiva C microcontroller that serves as a precise system timer. It can generate accurate millisecond delays using the wait_ms() function and produce periodic interrupts at configurable intervals.
The wait_ms() function purposes

- *Ensures* stable power-up and synchronization of peripherals (LCD, GPS module) before sending commands.
- Gives users time to read messages (e.g., "Nearest Place").
- Prevents flickering during screen updates
- Limits how often the GPS module is queried to balance responsiveness and power efficiency.
- Ensure smooth transitions between modes (e.g., "State 1" → "State 2") without overwhelming the user.
- Creates visual feedback (e.g., loading dots) during GPS acquisition.

TIMER

TIMER.h – Header File

contains function declarations and variables needed to manage **timer** using the Timer0A of the TM4C123 microcontroller.

1-Global Variables (extern):

- `seconds`: keeps track of elapsed seconds.
- `minutes`: keep track of elapsed minutes.
- `timer_running:` a flag that shows whether the timer is currently active.

2-Function Declarations:

- `Timer0A_Init:` initializes Timer0A (sets it up to count time).
- `Timer_Start:` starts or resumes the timer.
- `Timer_Pause:` pauses or stops the timer.
- `Timer_Reset` resets the timer values to zero.

Timer.c - Source File

Implements a simple digital timer that keeps track of time in seconds and minutes using the Timer0A module on the TM4C123 microcontroller.
This file works closely with the LCD display to show the updated time every second. It also provides functions to control the timer such as starting, pausing, and resetting.

Functions used:

1-The `Timer0A_Init:` function is responsible for setting up the timer. It enables the Timer0 clock, configures the timer to operate in 32-bit periodic mode, and sets it to overflow every 1 second (based on a 16 MHz clock). It also enables the timer interrupt so that a special function runs every second when the timer ticks.

2-The `Timer_Start`: is used to resume the timer if it was paused, by setting the `timer_running` flag to 1 and enabling Timer0A.

3-`Timer_Pause`: stops the timer by disabling Timer0A and setting the flag to 0.

4- **The `Timer_Reset`:** sets both `minutes` and `seconds` back to 0, essentially restarting the timer from the beginning.

5-`TIMER0A_Handler`:

- It first clears the interrupt flag, If the timer is running, it increments the `seconds` variable. Once `seconds` reaches 60, it resets to 0 and increases the `minutes` counter.
- The code also ensures that the timer resets completely after reaching 99 minutes and 59 seconds, preventing it from displaying more than two-digit minutes.
- To display the time, the function converts the current values of `minutes` and `seconds` into strings. It sends these strings to the LCD screen in the format "MM: SS".
  where: MM represents minutes (00 to 99)
  SS represents seconds (00 to 59)
- It also adds leading zeroes to maintain a consistent time format for example ("09:05" instead of "9:5")

UART

UART stands for Universal Asynchronous Receiver/Transmitter
It is a hardware communication protocol commonly found in microcontrollers, such as the Tiva C board.

UART Use Cases:

Receiving Data from GPS:
UART is used to receive live coordinate data from the GPS module, establishing a communication link between the GPS and the TM4C123G microcontroller.
Sending Data to LCD:
UART is used to send location information from the TM4C123G to the LCD screen, enabling the display of nearby landmark names based on the received GPS data.

Files:

UART.h - Header File: provides function declarations and definitions related to UART

UART.c - Source File: It defines how UART0, UART1, and UART2 are initialized and how they send and receive data.

Functions used:

1-Initialization Functions (`UARTx_Init`)

`UART0_Init()`, `UART1_Init()`, and `UART2_Init()` perform similar steps:

- **Enable the clock** for the corresponding UART module and its associated GPIO port.
- **Wait until the peripheral is ready** to be configured.
- **Disable the UART module temporarily** during configuration to avoid errors.
- **Set the baud rate** the code sets the baud rate to 9600
- **Configure data frame settings** (8-bit word length, FIFO enabled).
- **Enable UART transmit (TX) and receive (RX)**
- **Set GPIO alternate functions** to allow UART pins to work properly (for example PA0 and PA1 for UART0).

2-Sending and Receiving a Single Character

Each UART module includes:

1-`UARTx_CharTX`:

- Waits until the UART transmit buffer is not full.
- Sends a single character through UART.

2-`UARTx_CharRX`:

- Waits until there is data in the receive buffer.
- returns one received character.

3. Sending a String

-`UARTx_StringTX (char *str):`

- Sends an entire string character-by-character until it reaches the null terminator (`'\0'`).

-`UART0_StringTX_withLen(char *str, int x):`

- Sends a string of a specific length x, without needing a null terminator.

4. Receiving a String

-`UARTx_StringRX(char *str, char stopCh):`

Receives characters into a string until it sees a specified stop character, then ends the string with a null terminator.

❖ HAL

GPS

The GPS sensor provides accurate location data in the form of latitude and longitude by receiving signals from GPS satellites. It communicates with the Tiva C microcontroller via a UART (Universal Asynchronous Receiver/Transmitter) interface. The GPS sensor sends location information as NMEA sentences, which the Tiva C can parse to extract coordinates. This data can then be displayed on an LCD screen or sent to a server. In our GPS tracking system project, we used this setup to display the distance between two GPS coordinates on an LCD screen.

Files:

The module consists of two files.

GPS.h: Header file containing declarations and constants, EARTH_RADIUS: The average radius of Earth (6371000.0 meters) used for Distance Calculation

The distance is computed using the Haversine formula:

distance = 2 * R * asin(sqrt(sin²(Δlat/2) + cos(lat1) *cos(lat2) *sin²(Δlong/2)))

GPS.c: Implementation file containing the actual code, Coordinate_fixed_places: A 2D array storing latitude and longitude coordinates of locations

Place	Latitude	Longitude
Hall A, B	3003.8531	3116.8222
Hall C & D	3003.8219	3116.8358
Credit	3003.8076	3116.6998
Main Building	3003.8904	3116.7280
Luban	3003.7884	3116.7706

The latitude and longitude are extracted from the GPGGA NMEA sentence.

Functions Used:

1. is_match(): Helper function to identify GPS message headers
2. readGPS(): Reads and parses GPS data from UART
3. ToDegree(): Converts GPS coordinates to decimal degrees
4. ToRad (): Converts degrees to radians
5. GPS_getDistance(): Calculates distance between two points using Haversine formula
6. GPS_getNearestPlace(): Finds the nearest predefined location
7. GPS_get_Distance_from_SELECTED_Place(): Calculates distance to a specific predefined location.

The implementation of GPS-based distance calculation enables efficient real-time geolocation processing. The integration of predefined locations enhances precision in determining a user's closest point of interest, making it valuable for navigation applications.

LCD

In our GPS tracking system project, we implemented a real-time display feature that shows both the distance between two GPS coordinates and the travel time on an LCD screen.

Distance Display:

- The system calculates the distance between the starting point and destination and displays the distance in meters with clear numerical formatting, shows progress updates as the user moves toward the destination.

Time Tracking: The microcontroller records time from the start of the journey

-three operating mode: the system toggles between modes using a button with the active mode always visible on the lcd.

By combining distance measurement with time tracking on a single LCD display, we've created an efficient navigation aid that enhances situational awareness during travel.

LCD.h File:

This file contains the function declarations for controlling the LCD screen. These functions allow the program to initialize the LCD, send commands or data, and display text

LCD.c File:

The file contains the implementation of all functions needed to control and operate an LCD screen using the TM4C123 microcontroller. It defines how to initialize the LCD, send data and commands, and display text. The LCD is operated in 8-bit mode using GPIO pins spread across ports A, B, D, and E.

Functions Used

- `LCD_Init`: Initializes the LCD with the required settings, sets all control and data pins as outputs and sends setup commands to prepare the LCD.
- `LCD_Command`: Sends control instructions to the LCD like clearing the screen or setting display modes.
- `LCD_Data`: Sends actual character data to be shown on the screen.
- `LCD_String`: Loops through a string and sends each character using LCD_Data(), Displays a full string (text) on the LCD.
- `LCD_WritePins()`: Internally sets all 8 data lines (D0 to D7) according to the byte that needs to be sent.

TIming: Delays (as wait_`ms(1)`) are added to give the LCD enough time to process data.

LED and Buzzer Alert System

To enhance user interaction and provide immediate feedback, an alert system was implemented using a LED and a buzzer connected to GPIO pins on Port A of the Tiva C board.

The function responsible for managing this system is:

void Alert_If_Close(int distance);

This function performs the following:

1. Monitors the calculated distance between the current GPS location and a predefined point.

2. If the distance is less than or equal to a defined threshold (ALERT_DISTANCE), the function: Activates the LED and the buzzer by setting the corresponding GPIO pins (LED_PIN and BUZZER_PIN) to HIGH.
3. If the distance becomes greater than the threshold: Deactivates both the LED and buzzer by clearing the GPIO pins.

This setup provides both visual (LED) and auditory (buzzer) indications when the user is approaching a target location.

SWITCH_BIN

These files are used to manage the state of a system (for example, changing modes on a device) using a push button or switch.

`void switch_state(void)`

- Increases the state value by 1 each time it's called.
- If the state goes above 2, it resets back to 0

`void select_state(char selected_state)`

Manually sets the state to a specific value.

❖ Services - Utilities

TM4C123.h

Hardware definitions (registers, bit masks) for working with peripherals.

bit_utilities.h

This header file defines macros that simplify bitwise operations on hardware registers. These operations are commonly used in embedded systems programming.

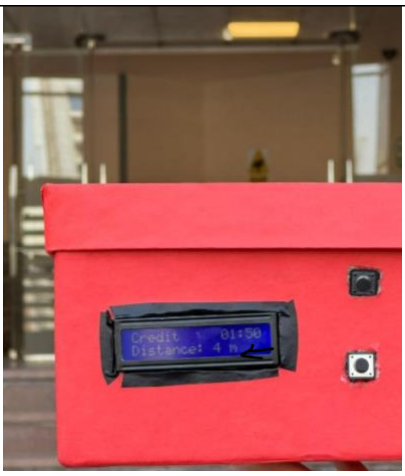
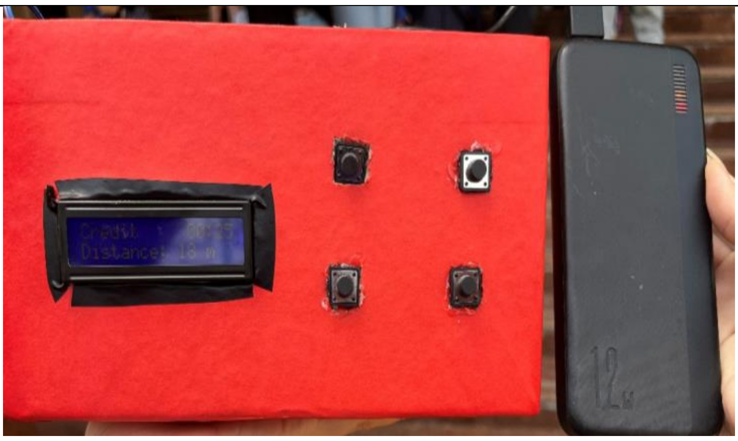
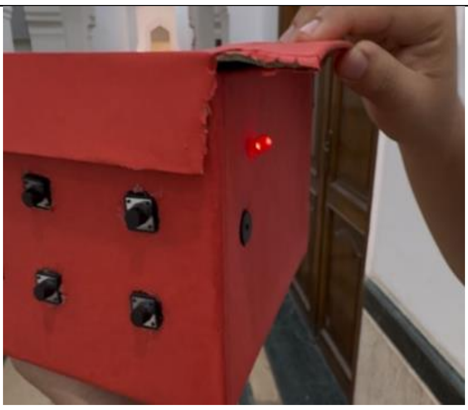
- READ_BIT(reg, bit): Reads the value of a specific bit.
- READ_REG(reg): Reads the lower 8 bits of a register.
- SET(reg, val): Sets multiple bits in a register.
- SET_BIT(reg, bit): Sets a single bit.
- CLEAR(reg, val): Clears multiple bits.
- CLEAR_BIT (reg, bit): Clears a single bit.

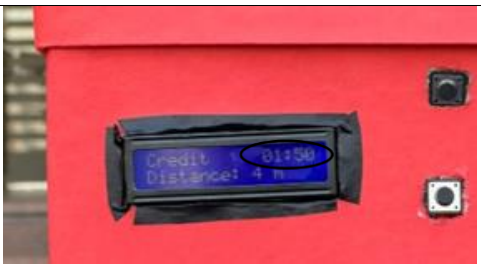
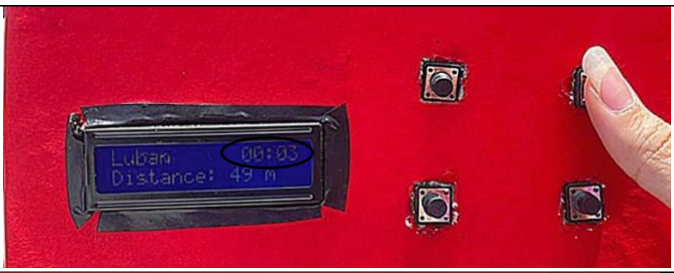
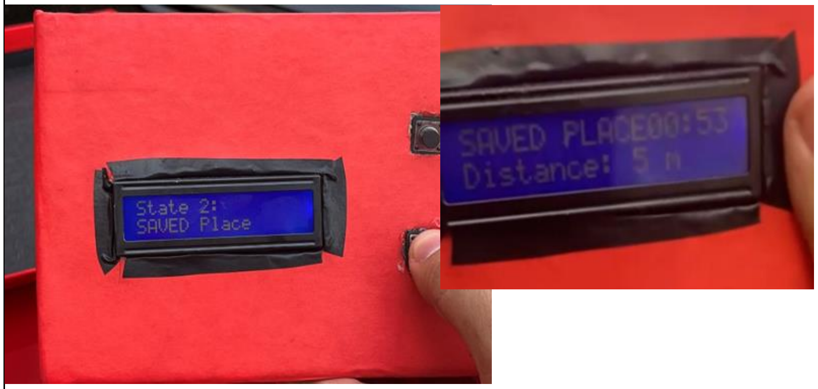
startup_TM4C123.s

Startup file with interrupt vector table and program entry point.

system_TM4C123.c

System initialization (e.g., clock setup) before entering main ().

## Bonus Features	
1. Display Distance to Nearest Place •Show the distance (in meter) on the LCD, helping users track progress toward their destination	
2. Power Source (power bank) Enable portable operation,making the device usable anywhere without fixed power ouylets and provide flexability for applications.	
3. LED and Buzzer Alerts • When within 20 meters of destination • LED flashes and buzzer sounds • Ensure users don't miss their destination, even if they are not looking at the display.	
4. LCD Driver • Custom-built I2C driver for 16x2 display • The driver handles LCD initializations, cursor control	

5. **Timer System** • Tracks elapsed time since startup, useful for logging travel duration or system diagnostics. • Updated every second via interrupt	
6. **Reset Timer Button** • Clears time to zero on press	
7. **Three Operating Modes** 1. **Mode 1 – Nearest Place** • Reads current location • Calculates distance to each saved landmark • Displays the closest place and distance 2. **Mode 2 – Save & Track** ○ Saves user's current location ○ Displays distance from that point in future sessions	  State 2: SAVED Place SAVED PLACE00:53 Distance: 5 m

Mode 3 – Manual Selection • User cycles through locations manually • Distance to selected location is shown	
8. Structured Layered Code Design • Clearly separated MCAL, HAL, APP folders	
9. Control brightness of Lcd using a potentiometer	https://drive.google.com/file/d/1YNwFiOi-DdApQ55NKs--YrI6jd4Sds6O/view?usp=drivesdk

Testing and Validation

- System tested across multiple open-air spots
- Distance accuracy within ±4 m
- Alert triggered consistently when user approached landmark
- Modes changed reliably with debounce logic
- Timer reset worked without errors

Conclusion

In this project, we successfully designed and implemented an embedded GPS tracking system using the TM4C123G LaunchPad. The system accurately retrieves real-time coordinates from a GPS module, calculates the nearest predefined landmark (such as faculty halls), and displays the location information dynamically. Key achievements include:

- Efficient GPS Data Parsing: Successfully extracted and processed NMEA sentences to obtain valid latitude and longitude.

- Landmark Detection: Implemented a distance-based algorithm to identify and display the closest landmark from stored coordinates.

- Real-Time Updates: Ensured periodic location refreshes, enabling smooth tracking during motion.

Overall, this project strengthened our skills in embedded C programming, hardware interfacing, and teamwork, meeting all functional requirements while laying a foundation for further advancements

References

- Tiva C TM4C123GH6PM Datasheet
- NEO-6M GPS Datasheet
- TivaWare Peripheral Driver Library Documentation