MIT-WPU

॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

Dr. Vishwanath Karad
MIT WORLD PEACE
UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

T. Y. B. Tech (ECE)

Trimester: VI **Subject:** ESD & RTOS

Name: Satej Zunjarrao **Division: B**

Roll No: PB-30 **Batch: B3**

Experiment No: 09

Name of the Experiment: Simple multitasking application with µCOS II RTOS (Use minimum 3 tasks)

Performed on:

Submitted on:

Marks	**Teacher's Signature with date**

Aim: Write Embedded C program for simple multitasking application with µCOS II RTOS on LPC2148 (Use minimum 3 tasks).

Part List:

- Educational practice board for ARM7 (EPBARM7)
- +9V Power supply
- USB A to B type cable
- PC
- Eclipse IDE
- Flash Magic Utility

Hardware Connection:

Connect USB A to B Type cable between PL3 connector of EPBARM7 board and PC.

Procedure:

Included Files:

HEADER FILES	**SOURCE FILES**
lcd.h	lcd.c
uart.h	uart.c
U-COSII folder	

Steps to create project and program compilation:

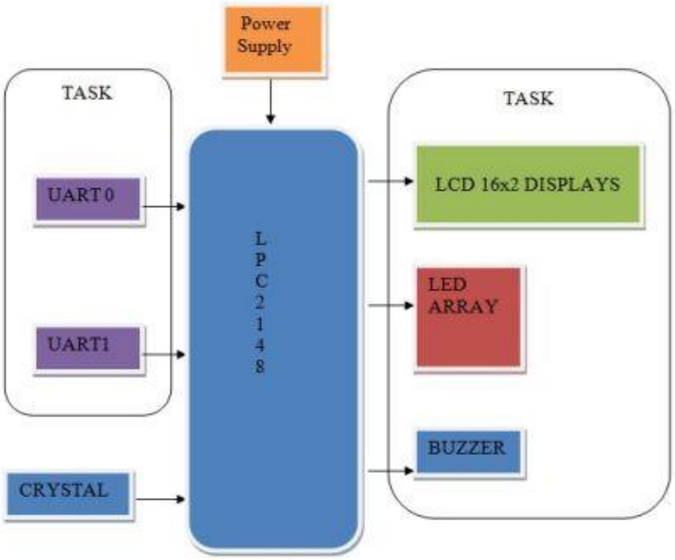
Steps:

- Open Eclipse.exe.
- Now browse to the ARM7 Workspace. Click OK to continue.
- Click File > **Import…**, to import uCosII_ Template.
- Click General > Existing Projects into Workspace and click Next.
- First Select Root Directory of your uCosII_ Template. Then select "Copy projects into workspace" check box and Click Finish.
- Every time you import a project make sure to rename it. So **Right Click Project > Rename** or press **F2** while selecting project to rename it.
- Go to File> New > Source File and you will see New Source File Wizard. Enter Source File name (For example **main.c**) then Click Finish.
- Write your code and then save your Files.
- Copy necessary .c and .h files to your local project folder. They will be added to your project in Eclipse IDE

Steps to use hardware:

- Connect 9V DC Power supply to the educational practice board for EPBARM7
- Connect the board with the USB port of the PC using the USB A to B type cable.
- Using the RUN/PROGRAM mode selection switch, set the board in the program mode. This will be indicated by the red LED.
- Apply Reset condition by pressing the RESET switch to ensure proper communication.
- Using download tool (Flash Magic) download the .HEX file to the target board.
- Open hyper terminal and set the baudrate 9600.
- Using the RUN/PROGRAM mode selection switch, set the board in the run mode. This will be indicated by the green LED and apply reset to execute the program.

Interfacing Diagram:



µCOS II functions used: (Detailed Description)

OSInit() - **Initializes the µC/OS-II kernel.**

OSTaskCreate() - **Creates a new task in the µC/OS-II system.**

OSStart() - **Starts the µC/OS-II kernel to begin multitasking.**

OSTimeDlyHMSM() - **Delays the execution of the current task for a specified amount of time. The arguments for this function are hours, minutes, seconds, and milliseconds.**

uprintf() - **A custom function that sends a formatted string to the UART0.**

These functions are used to create and manage tasks in the system, delay task execution for a specified period of time, and print output to the LCD and UART0.

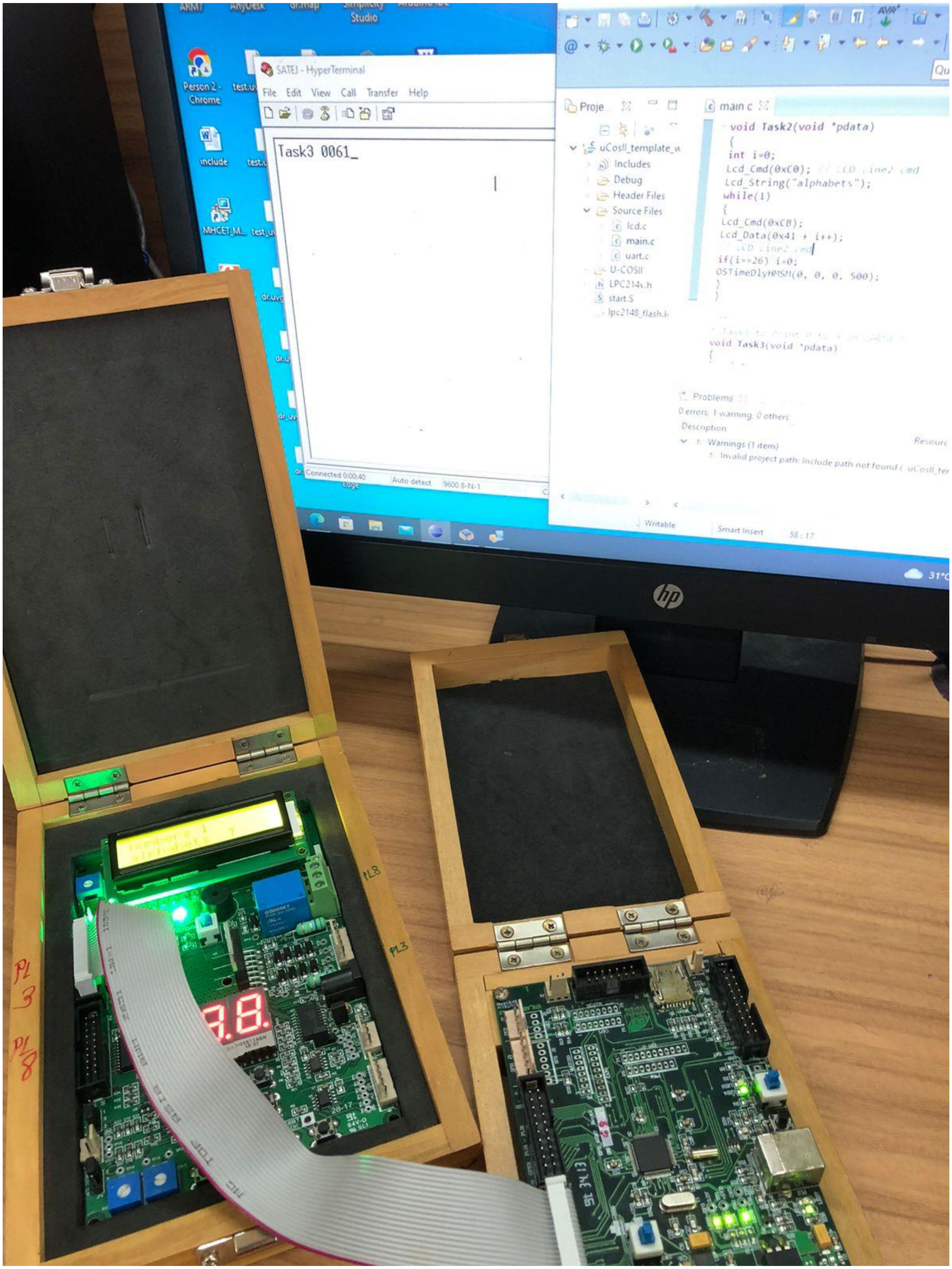
Program:

```
#include "includes.h"

#include "edutech.h"

#include "uart.h"

#include "lcd.h"
```

```
#define UART_DEBUG   0




/* Task1 Stack */
OS_STK Task1Stack[100];
void Task1(void *pdata);

/* Task2 Stack */
OS_STK Task2Stack[100];
void Task2(void *pdata);

/* Task3 Stack */
OS_STK Task3Stack[100];
void Task3(void *pdata);



/* Main Program */
int  main (void)
{
      timer_init();    // initialize OS Timer Tick

      OSInit();        // Initialize uC/OS-II




      OSTaskCreate(Task1, (void *)0, &Task1Stack[99], 1);      // Create
task1
      OSTaskCreate(Task2, (void *)0, &Task2Stack[99], 2);      // Create
task2
```

```
    OSTaskCreate(Task3, (void *)0, &Task3Stack[99], 3);      // Create
task3


    /* start the multitasking process which lets uC/OS-II manages the
task that you have created */
    OSStart();
    return 0;
}



/* Task Definition */
/**
 * Task1 to Print A to Z on LCD line1
 */
void Task1(void *pdata)
{
    int i=0;
    Lcd_Init();                 // Initialize LCD in 8bit mode
    Lcd_Cmd(0x01);              // LCD clear cmd
    Lcd_Cmd(0x80);             // LCD Line1 cmd
    Lcd_String("T1 ");

    while(1)
    {
        Lcd_Cmd(0x83);              // LCD Line1 cmd
        Lcd_Data(0x41 + i++);
        if(i==26) i=0;
        OSTimeDlyHMSM(0, 0, 0, 500); //delay 500ms
    }
}
```

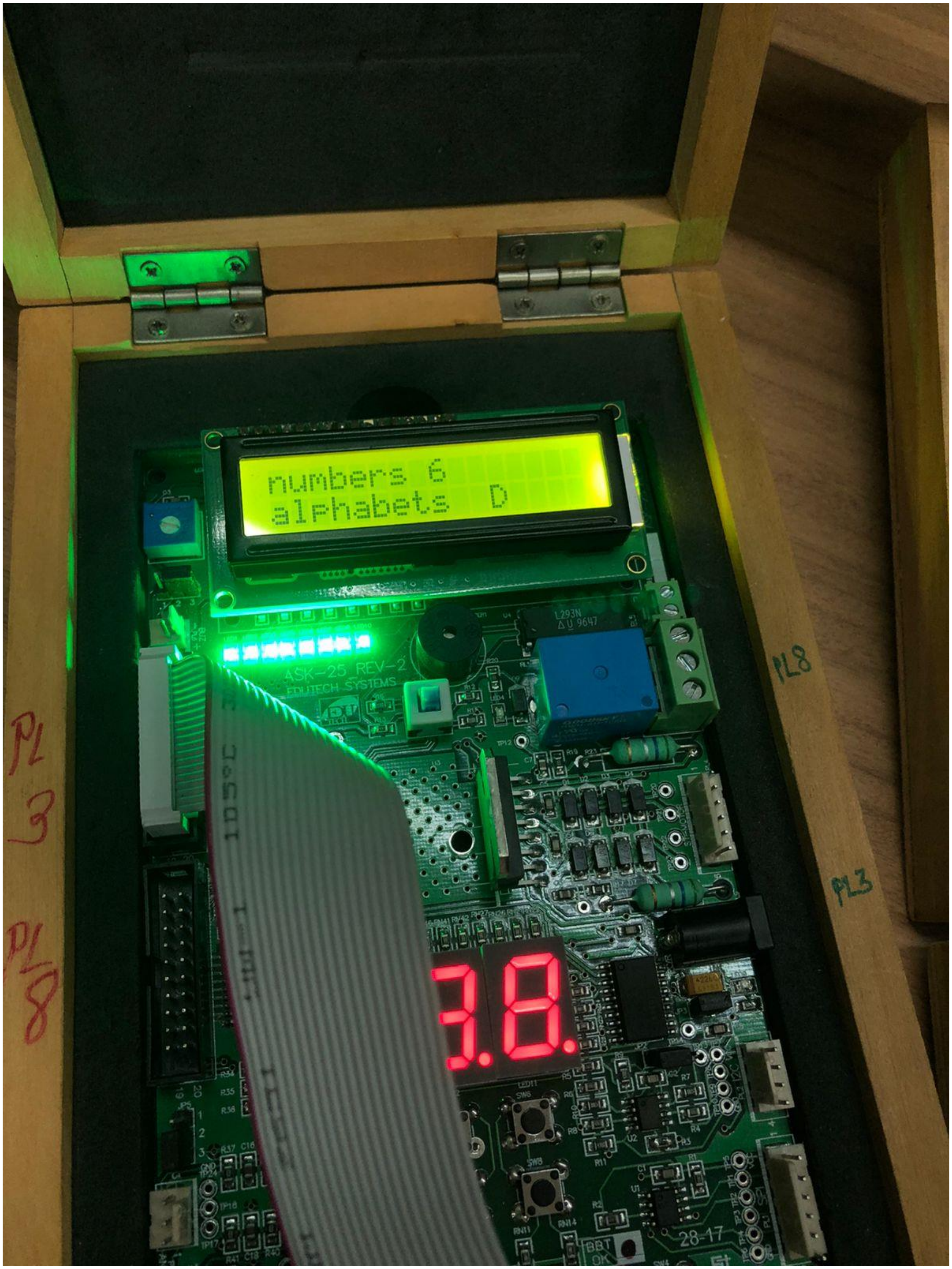
```
/**
 * Task2 to Print 0 to 9 on LCD line2
 */
void Task2(void *pdata)
{
      int i=0;
//    Lcd_Init();                  // Initialize LCD in 8bit mode
      Lcd_Cmd(0xC0);             // LCD Line2 cmd
      Lcd_String("T2 ");

      while(1)
      {
            Lcd_Cmd(0xC3);             // LCD Line2 cmd
            Lcd_Data(0x30 + i++);
            if(i==10) i=0;
            OSTimeDlyHMSM(0, 0, 0, 500); // Delay 1s
      }
}


/**
 * Task3 to Print 0 to 9999 on UART0
 */
void Task3(void *pdata)
{
      int i=0;
      Uart0_Init(9600);      // Initialize UART
      while(1)
      {
        uprintf("\x1b[1;1HTask3 %d04",i++);        // print on UART0
        if(i==9999) i=0;
        OSTimeDlyHMSM(0, 0, 0, 700); // Delay 700ms
    }
}
```



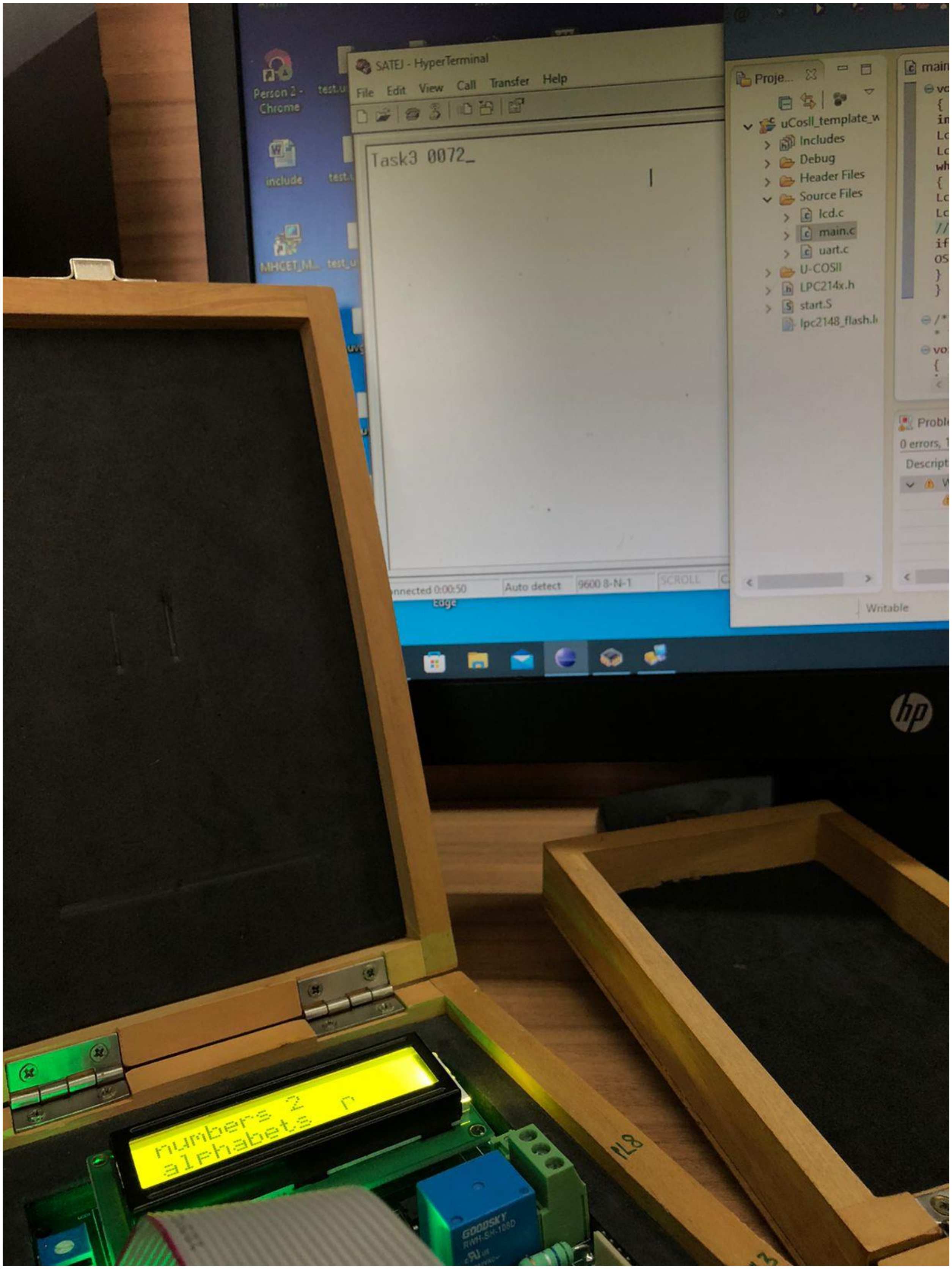
```
Task3 0061_
```

```
void Task2(void *pdata)
 {
 int i=0;
 Lcd_Cmd(0xC0); // LCD Line2 cmd
 Lcd_String("alphabets");
 while(1)
 {
 Lcd_Cmd(0xCB);
 Lcd_Data(0x41 + i++);
 // LCD Line2 cmd
 if(i==26) i=0;
 OSTimeDlyHMSM(0, 0, 0, 500);
 }
 }

void Task3(void *pdata)
{
```

```
Problems
0 errors, 1 warning, 0 others
Description
Warnings (1 item)
Invalid project path: Include path not found ( uCosII_ter
```



```
numbers 6
alphabets   D
```



```
Task3 0072_
```

```
numbers 2   r
alphabets
```

Conclusion:

In conclusion, implementing an RTOS on the LPC2148 microcontroller has proven to be a highly effective solution for multitasking applications. By using an RTOS, we were able to efficiently manage and schedule multiple tasks with different priorities while maintaining real-time performance. This experiment demonstrated the power of using an RTOS for embedded systems development and highlights the importance of choosing the right RTOS for a given application. The LPC2148 microcontroller's compatibility with a wide range of RTOS options makes it a highly versatile choice for embedded systems projects.