Fifth Semester

19CSE303 EMBEDDED SYSTEMS

Case Study

<< Weather monitoring system, via I2C, Bluetooth >>

Team Members:

Registration No	Name	Email ID
CB.EN.U4CSE21001	Abhinav R	Cb.en.u4cse21001@cb.students.amrita.edu
CB.EN.U4CSE21015	Dharaneesh P	Cb.en.u4cse21015@cb.students.amrita.edu
CB.EN.U4CSE21021	Hariharan Arul	Cb.en.u4cse21021@cb.students.amrita.edu
CB.EN.U4CSE21050	Niranjan S	Cb.en.u4cse21050@cb.students.amrita.edu
CB.EN.U4CSE21053	Samya Rebecca	Cb.en.u4cse21053@cb.students.amrita.edu

AIM:

Design and implement a comprehensive weather monitoring system that utilizes ADC for efficient sensor integration, I2C and OLED for RealTime Data Monitoring, and incorporates Bluetooth technology to enable wireless transmission of weather Data, fostering a reliable and user-friendly solution for real-time weather data acquisition and analysis.

Motivation:

There is a heavy demand for remote realtime Temperature and Humidity monitoring systems in multiple fields.

Medical Instruments: Certain medical instruments need to be stored at specific temperatures and humidity levels.

Food Processing: Maintaining the right temperature and humidity is crucial for food safety and quality.

Warehouses and Cold Storages: Warehouses storing temperature-sensitive products like chemicals, fruits, vegetables, food, medicines, etc., need these systems.

Agriculture: Monitoring temperature and humidity can help optimize crop growth and prevent disease.

Data Centers: These facilities require strict climate control to prevent overheating and ensure optimal performance.

Open challenges and problems faced in our topic:

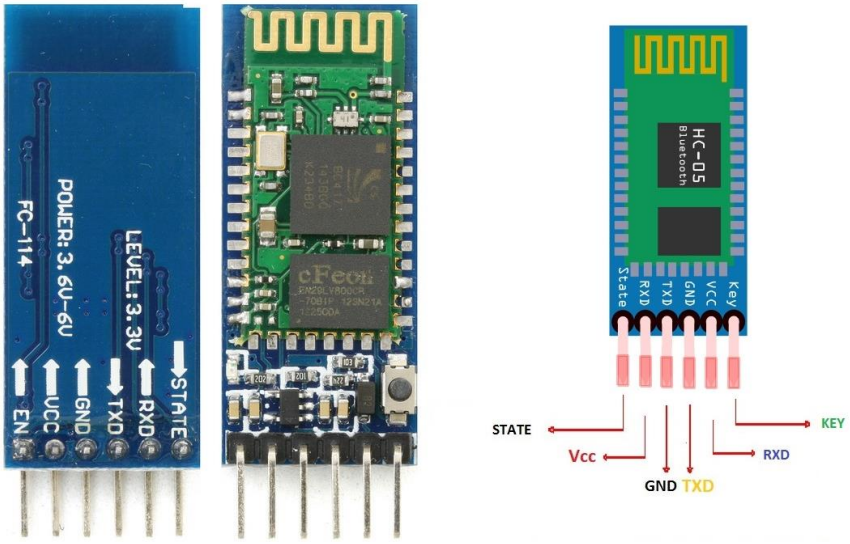
Sensor Performance: Sensor performance values can vary widely, and basic testing by manufacturers is often lacking. This can lead to inaccurate readings and unreliable data.

Power Consumption: There is a need to keep power consumptions by these Real Time Monitoring Systems as minimal as possible.

Data Transmission: Ensuring effective long-distance data transmission with minimal power consumption can be a challenge.

Components used along with their pictures and their pin diagrams:

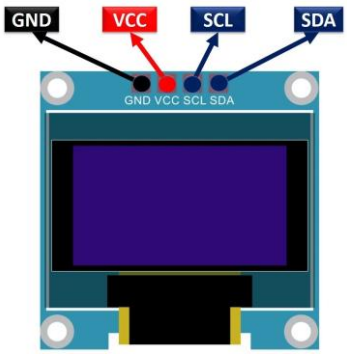
1. **HC-05 Bluetooth Transmitter and Receiver**



HC-05 is a two-way full-duplex Bluetooth to serial module. In other words, it uses UART serial communication to transmit and receive data packets through a classic Bluetooth standard. Its CSR Bluecore 04-External single chip is already configured to communicate with other Bluetooth devices through serial communication.

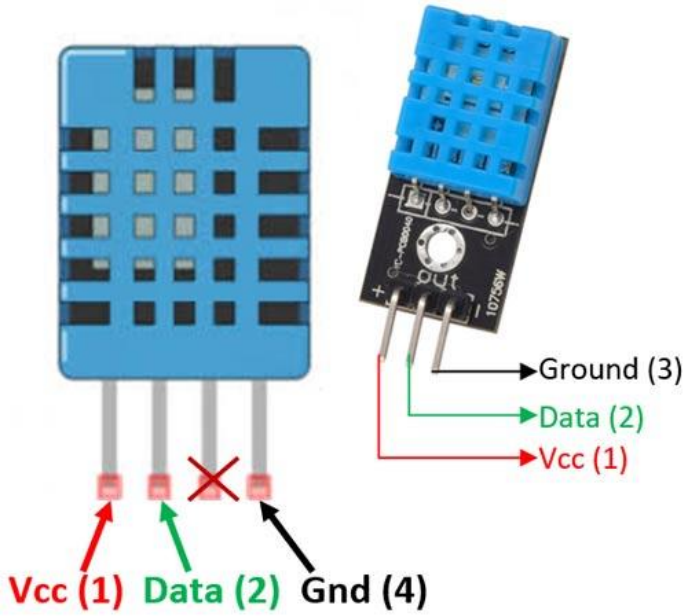
The default baud rate set for HC-05 is 9600.

2. **SSD1306 0.96 inch I2C OLED**



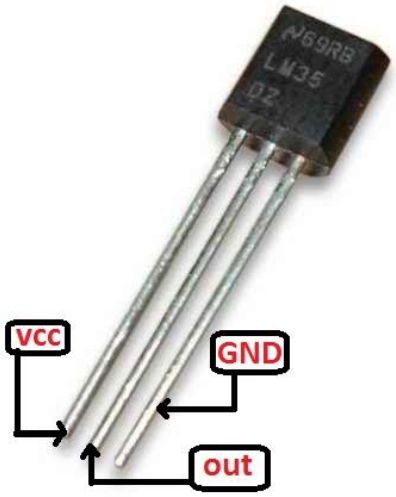
This is a I2C based OLED display having size of 128×64 pixels. SSD1306 CMOS OLED driver controller is used to communicate with microcontrollers, such as TM4C123 Tiva Launchpad, using either SPI or I2C communication. SCL and SDA are the serial clock and serial data pins respectively. They connect with I2C pins of microcontrollers (TM4C123) to perform I2C communication.

3. **DHT11 Temperature and Humidity Sensor**



DHT11 sensor consists of a capacitive humidity sensing element and a thermistor for sensing temperature. The humidity sensing capacitor has two electrodes with a moisture holding substrate as a dielectric between them. Change in the capacitance value occurs with the change in humidity levels. The IC measure, process this changed resistance values and change them into digital form. The temperature range of DHT11 is from 0 to 50 degree Celsius with a 2-degree accuracy. Humidity range of this sensor is from 20 to 80% with 5% accuracy. The sampling rate of this sensor is 1Hz.

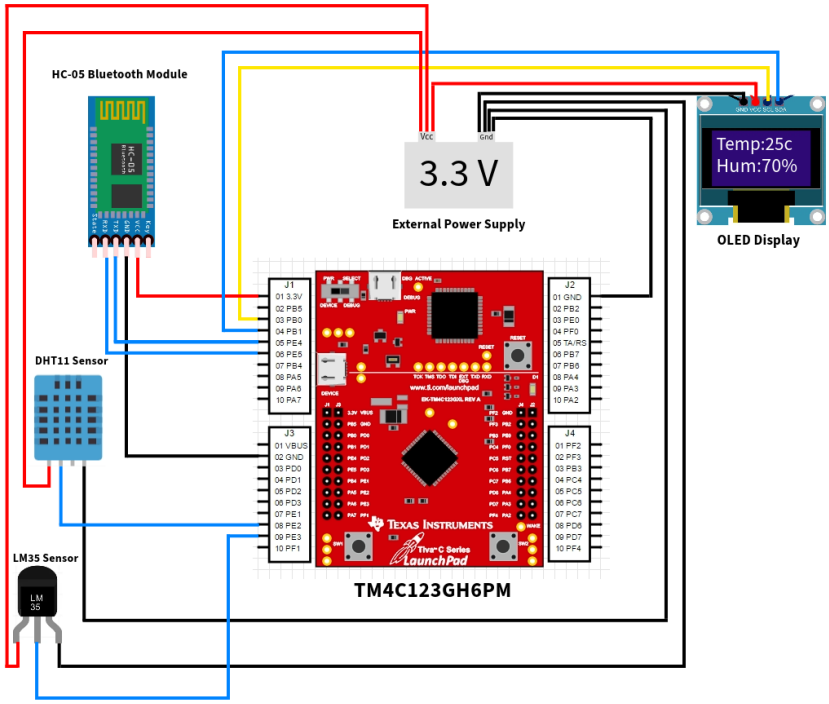
4. LM35 Analog Temperature Sensor



LM35 is a precession Integrated circuit Temperature sensor, whose output voltage varies, based on the temperature around it. It is a small and cheap IC

which can be used to measure temperature anywhere between -55°C to 150°C. It can easily be interfaced with any Microcontroller that has ADC function

Architecture diagram with labelling:

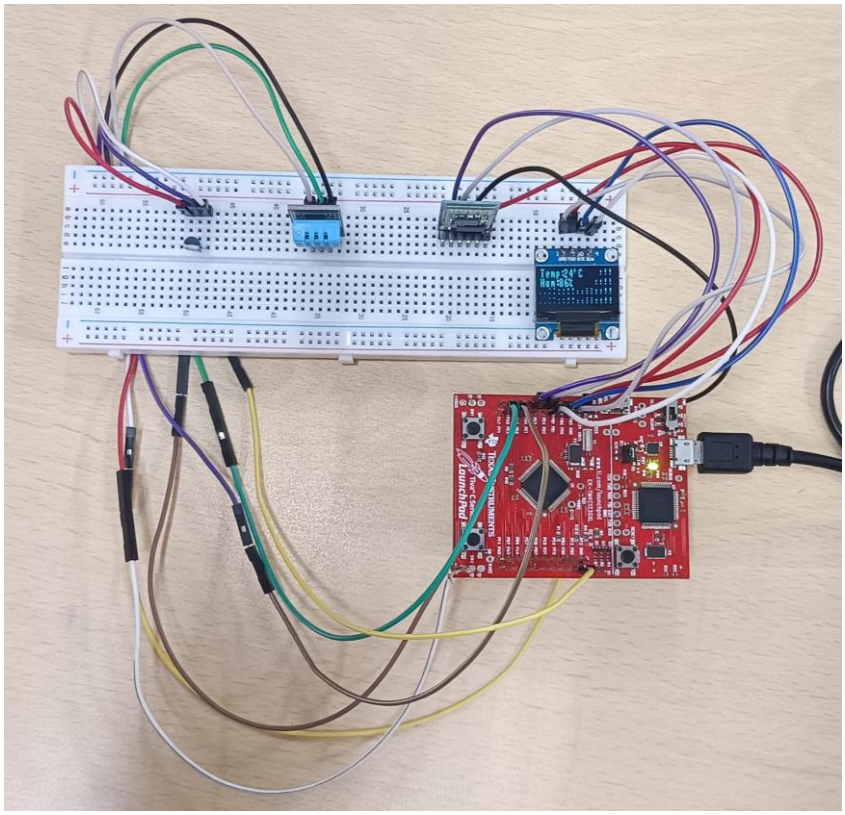


NOTE:

-PB0 and PB1 are mismatched to PD0 and PD1 on board printing so while connecting do connect SCL of OLED to PD0 of Launchpad and SDA of OLED to PD1 of launchpad. No change in code is required.

-Bluetooth module works only if its VCC is connected to VBUS of Launchpad which is to the immediate right of +3.3V pin of Tiva board.

Physical implementation and demo pictures, screenshots



Code Breakdown with key snippets and register configurations:

1. **Configuring PE3 for DHT11 as digital**

```
 SYSCTL->RCGCGPIO |= (1 << 4);   /* Enable Clock to GPIOE */
/* initialize PE3 for DHT11 data pin */
GPIOE->AFSEL |= (1 << 3);       /* enable alternate function */
GPIOE->DEN |= (1 << 3);         /* enable digital function */
GPIOE->DIR |= (1 << 3);         /* set as output initially */
GPIOE->DATA |= (1 << 3);        /* pull high initially */
```

2. **Configuring PE2 for LM35 as analog**

```
SYSCTL->RCGCADC |= (1<<0);    /* AD0 clock enable*/

   /* initialize PE2 for AIN0 input  */
   GPIOE->AFSEL |= (1<<2);      /* enable alternate function */
   GPIOE->DEN &= ~(1<<2);       /* disable digital function */
   GPIOE->AMSEL |= (1<<2);      /* enable analog function */


   /* initialize sample sequencer3 */
   ADC0->ACTSS &= ~(1<<3);       /* disable SS3 during configuration */
   ADC0->EMUX &= ~0xF000;    /* software trigger conversion */
   ADC0->SSMUX3 = 0;        /* get input from channel 0 */
   ADC0->SSCTL3 |= (1<<1)|(1<<2);       /* take one sample at a time, set flag at
1st sample */
   ADC0->ACTSS |= (1<<3);        /* enable ADC0 sequencer 3 */
```

3. Getting the actual reading from both the sensors and processing them:

```
readDHT11(&humidity, &temperature_dht);


         ADC0->PSSI |= (1<<3);       /* Enable SS3 conversion or start sampling
data from AN0 */
while((ADC0->RIS & 8) == 0) ;   /* Wait untill sample conversion completed*/
adc_value = ADC0->SSFIFO3;
// Transform the raw adc reading to analog voltage
V_sensor = ((adc_value * 3.3 )/ 4095)*1000 ;
// Converting analog voltage to temperature reading in degree celscius
```

```
temperature = (V_sensor - 1200) / 16;
// Take average of both the temperature readings from DHT11 and LM35
temperature = (temperature + temperature_dht)/2;
```

4.Passing the variables temperature and humidity to be displayed in OLED:

```
OLED_YX(0, 0 );
        OLED_Write_String("Temp:" );
        OLED_YX(0, 5);
        OLED_Write_Integer((int)(temperature));

        OLED_YX(1, 0 );
        OLED_Write_String( "Hum:" );
        OLED_YX(1, 4);
        OLED_Write_Integer((int)(humidity));
```

5. Displaying in Bluetooth based on character recieved from mobile:

```
if((UART5->FR & (1<<4))  == 0){ /* if Rx FIFO not empty */
        char c = Bluetooth_Read();
// Bluetooth connection testing with red led on and off
        if( c=='A'){
           GPIOF->DATA |=(1<<1);
           Bluetooth_Write_String("RED LED ON\n");
        }
        else if( c=='B'){
           GPIOF->DATA &=~(1<<1);
```

```
            Bluetooth_Write_String("RED LED OFF\n");
          }
//Actual function
          else if( c=='T'){

            Bluetooth_Write_Temperature(temperature);
Bluetooth_Write_String("\n");

          }
          else if( c=='H'){

            Bluetooth_Write_Humidity(humidity);
Bluetooth_Write_String("\n");


          }
       }


  }
```

5. Implementation of custom DHT11 function to extract the relevant and correct info from DHT11 sensor output:

```
void readDHT11(uint8_t* humidity, uint8_t* temperature) {
   // Initialize variables to store the data
   uint8_t data[5] = {0};
```

```
    // Request data from the DHT11 sensor
    GPIOE->DIR |= (1 << 3);    // Set PE3 as output
    GPIOE->DATA &= ~(1 << 3);   // Pull the line low
    Delay_ms(18000);  // Delay for at least 18ms (DHT11 datasheet
requirement)
    GPIOE->DATA |= (1 << 3);    // Release the line
    Delay_ms(40);     // Wait for the DHT11 response

    // Change pin to input for receiving data
    GPIOE->DIR &= ~(1 << 3);

    // Wait for the sensor to respond
    while ((GPIOE->DATA & (1 << 3)) != 0);

    // Wait for the sensor to finish the first response (80us low)
    while ((GPIOE->DATA & (1 << 3)) == 0);
    Delay_ms(80);

    // Read data from the sensor
    for (int i = 0; i < 5; ++i) {
        // Read 8 bits for each data byte
        for (int j = 7; j >= 0; --j) {
            // Wait for a falling edge (start of bit)
            while ((GPIOE->DATA & (1 << 3)) == 0);
            // Wait for a rising edge (end of bit)
            while ((GPIOE->DATA & (1 << 3)) != 0);
            // Record the time to determine if it's a 0 or 1
```

```
            Delay_ms(40);
            if (GPIOE->DATA & (1 << 3)) {
                // High pulse is longer, consider it a 1
                data[i] |= (1 << j);
            }
        }
    }

    // Check the checksum
    if (data[0] + data[1] + data[2] + data[3] == data[4]) {
        // Data is valid
        *humidity = data[0];
        *temperature = data[2];
    } else {
        // Data is invalid
        *humidity = 0;
        *temperature = 0;
    }
}
```

Features and advantages, list of existing problems solved by our implementation

Features:

Floating-Point Unit (FPU):

The Cortex-M4 incorporates a dedicated Floating-Point Unit, enhancing the efficiency of floating-point arithmetic operations. This feature is especially advantageous in applications requiring intricate mathematical calculations, such as digital signal processing and control systems.

Digital Signal Processing (DSP):

The inclusion of DSP capabilities in the Cortex-M4 significantly improves computational efficiency. It accelerates signal-processing tasks across various applications, making it particularly beneficial in scenarios where rapid and efficient signal handling is crucial.

Advanced Interrupt Handling:

The Cortex-M4 supports nested vectored interrupt controllers (NVIC), allowing for prioritized and efficient handling of interrupts. This capability is vital in real-time applications, ensuring a timely response to external events.

Enhanced Performance:

With a 3-stage pipeline, branch prediction, and higher clock speeds, the Cortex-M4 delivers enhanced execution efficiency. These features collectively enable the processor to meet the demands of more complex and resource-intensive applications.

Harvard Bus Architecture with Unified Memory Space:

The Cortex-M4 employs a Harvard architecture with unified memory space, where instructions and data share the same address space. This design enhances memory access efficiency and simplifies programming.

32-Bit Addressing:

Supporting 32-bit addressing, the Cortex-M4 allows access to a vast 4GB memory space, accommodating the needs of memory-intensive applications.

Bit Band Feature:

The processor incorporates a Bit Band feature, enabling efficient bit-data accesses in two specific memory regions. This feature streamlines the manipulation of individual bits in memory, enhancing overall data handling.

Single Instruction Multiple Data (SIMD) Operations:

The Cortex-M4 supports SIMD operations, enabling parallel processing of multiple data elements with a single instruction. This feature enhances performance in tasks that benefit from data-level parallelism.

Memory Protection Unit (MPU):

An optional Memory Protection Unit (MPU) provides advanced memory protection features, including programmable memory and access permission control. This allows for specific memory regions to be protected or controlled, such as disabling the execution of code located in SRAM.

JTAG Standard:

Named after the Joint Test Action Group, the JTAG standard is supported, providing an industry-standard interface for verifying designs and testing printed circuit boards post-manufacture. This ensures robustness and reliability in the validation and testing processes.

Other features:

I2C Sensor Integration: Utilizing I2C facilitates the integration of diverse sensors, such as temperature, humidity, and atmospheric pressure, providing comprehensive weather data.

Bluetooth Connectivity: Incorporating Bluetooth technology enables wireless communication, allowing users to access and control the system remotely, fostering increased flexibility and convenience.

Real-time Data: The system provides real-time weather data, allowing for timely decision-making and analysis.

User-friendly Interface: A user-friendly interface enhances accessibility, making it easier for users to interact with the system, visualize data, and configure settings.

Scalability: The modular design allows for the addition of new sensors or features, making the system adaptable to evolving weather monitoring needs.

Advantages:

Versatility: The system can be applied in various settings, including agriculture, research, and home automation, due to its flexible and versatile nature.

Wireless Operation: Bluetooth connectivity eliminates the need for physical connections, reducing clutter and providing greater freedom in system placement.

Remote Accessibility: Users can monitor and control the system remotely, improving convenience and accessibility.

Efficient Resource Usage: I2C communication ensures accurate data transmission between the board and the microcontroller in just 2 wires, thus efficiently using the resources

Integration Potential: The system can be integrated into larger networks or smart home systems, contributing to a more connected and intelligent environment

Enhanced Computational Power:

The inclusion of a Floating-Point Unit (FPU) accelerates complex mathematical operations, improving the overall computational power of the system.

Optimized Signal Processing:

The Digital Signal Processing (DSP) capabilities boost efficiency in handling signal-processing tasks, making it highly suitable for applications requiring real-time and high-performance signal processing.

Efficient Interrupt Handling:

Advanced interrupt handling, with support for nested vectored interrupt controllers (NVIC), ensures timely and prioritized response to external events. This is crucial for real-time applications where responsiveness is paramount.

Improved Execution Efficiency:

Features such as a 3-stage pipeline, branch prediction, and higher clock speeds contribute to enhanced execution efficiency. This results in better performance for demanding applications and tasks.

Memory Protection and Security:

The optional Memory Protection Unit (MPU) adds a layer of security by providing memory protection features.

Problems Solved:

1. **Data Accessibility:** Traditional weather monitoring systems may lack remote access. Our implementation addresses this by incorporating Bluetooth, allowing users to access data from a distance.

2. **Limited Flexibility:** Some weather monitoring systems may lack adaptability. The modular and scalable design of our system addresses this by allowing the addition of new sensors or features as needed.
3. **Complex User Interaction:** Systems with complicated interfaces may deter users. Our user-friendly interface simplifies interaction, making the system accessible to a broader audience.
4. **Complex Mathematical Operations:** The FPU in Cortex-M4 accelerates floating-point arithmetic, addressing the challenge of performing intricate mathematical calculations efficiently.
5. **Inefficient Signal Processing:** DSP features in Cortex-M4 improve computational efficiency, making it suitable for applications requiring real-time and high-performance signal processing.
6. **Interrupt Handling in Real-Time Systems**: Advanced interrupt handling with nested vectored interrupt controllers (NVIC) in Cortex-M4 allows for prioritized and efficient interrupt handling, critical for real-time systems.
7. **Execution Inefficiency:** Cortex-M4's features, such as a 3-stage pipeline, branch prediction, and higher clock speeds, contribute to enhanced execution efficiency, addressing the challenge of performance limitations**.**
8. **Memory Protection and Security:** The optional Memory Protection Unit (MPU) provides features like programmable memory and access permission control, addressing concerns related to memory security.

Conclusion:

This project integrates I2C communication for sensor connectivity, Bluetooth-UART for wireless communication, and OLED display for real-time data visualization. Using an ADC, it accurately measures temperature from LM35 and humidity from DHT11 sensors. The Cortex-M4 processor efficiently handles complex calculations and interrupt handling, making it suitable for real-time applications. This comprehensive weather monitoring system enhances accessibility, efficiency, and flexibility, providing users with a reliable, wireless solution for monitoring and analyzing real-time weather data.

Future work:

Remote weather monitoring and control system system via wifi and response with DC motor.

References:

- Furber SB. ARM system-on-chip architecture. pearson Education; 2000.
- Martin T. The Insider's guide to the Philips ARM7-based microcontrollers. Coventry, Hitex, UK, Ltd. 2005.
- **https://microcontrollerslab.com/oled-interfacing-with-tm4c123g-display-texts-and-graphics/**
- **https://microcontrollerslab.com/hc-05-bluetooth-interfacing-tm4c123g-tiva-launchpad-keil-uvision/**
- **https://microcontrollerslab.com/adc-tm4c123g-tiva-c-launchpad-measure-analog-voltage-signal/**

Project Code:

project_code.txt

Working Video:

EmbeddedSystems_Evaluation.mp4