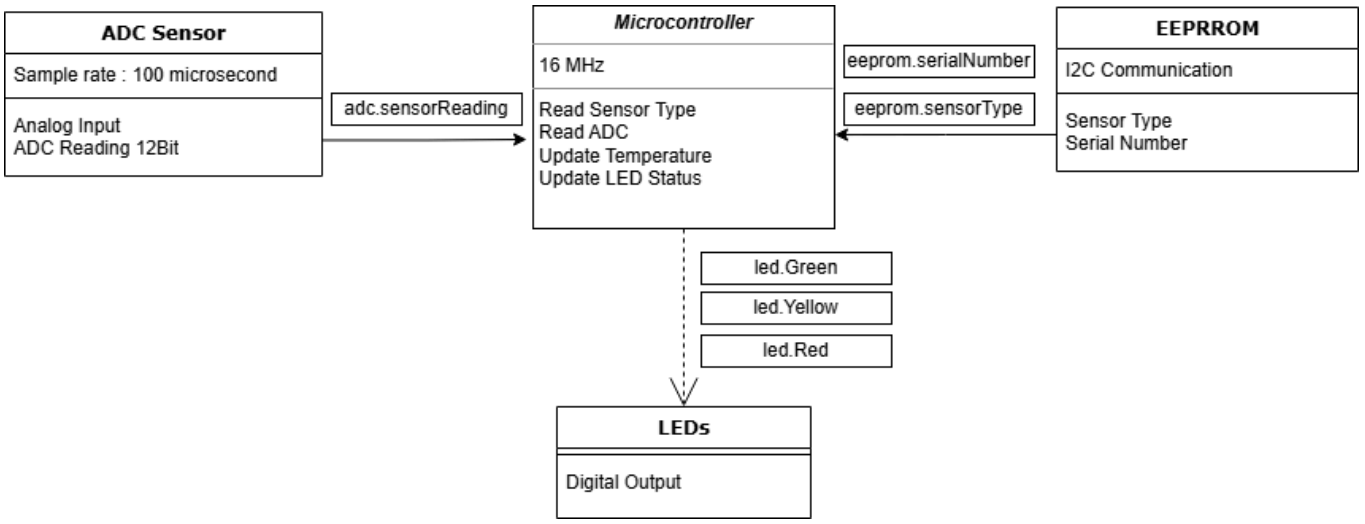
EMBEDDED TASK ASSIGNMENT

System Setup



Proposed system structure contains 1 ADC sensor (type can be changed) as an analog input and EEPROM data as a digital input. After processing inputs with an example 16mHz microcontroller led status is updated as a digital output.

For simulating ADC reading , random number generator is used according to sensor type.
12Bit ADC (range : 0-4095) is chosen. For implementing 100 micro second reading rate, ISR interrupt and timer/counter function are implemented. The setting commands can be different on each microcontroller type. Calculations are done according to 16 MHz microcontroller clock rate.

```
void adcInit(void) {
  printf("Mock ADC initialized for virtual channel./n");
  printf("Clearing ADCAN0 interrupt flag./n");
  printf(" Set Timer/Counter1 to CTC mode (Clear Timer on Compare Match)./n");
  // Set the compare value to achieve approximately 100us with a 16MHz clock
  // Use a prescaler of 64.
  // 16000000 Hz / 64 = 250000 Hz
  // 1 / 250000 Hz = 4 microseconds per tick
  // To get 100us, we need 100 / 4 = 25 ticks.

  printf("Initialize counter. Enable Timer/Counter1 Interrupt.Set the prescaler to 64/n");

  // Seed the random number generator
  srand(time(NULL));
  // Initialize the history with some random values
  for (int i = 0; i < ADC_HISTORY_SIZE; i++) {
    adc_history[i] = rand() % 4096; // Simulate a 12-bit ADC (0-4095)
  }
  printf("Setting ADCAN0 interrupt flag./n");
};
```

```
void adcRead(void) {
    uint16_t simulatedValue;
    // Simulate reading a slightly changing value
    if (eeprom.sensorType == 1)
        simulatedValue = rand() % 100 + 50; // Simulate a value around 50-150
    if (eeprom.sensorType == 2)
        simulatedValue = rand() % 1000 + 500; // Simulate a value around 500-1500

    printf("Read the ADC value from the ADCBUF./n");
    printf("clear the ADCAN0 interrupt flag./n");
    printf("Mock ADC reading  , simulated_value:%d \n", simulatedValue);

    // Update the history
    adc_history[history_index] = simulatedValue;
    history_index = (history_index + 1) % ADC_HISTORY_SIZE;

    adc.sensorReading=simulatedValue;
    printf("ADC process has finished. Real Time ISR is triggered./n");
     // Clearing ADC0 interrupt.
     // Enabling ADC0 interrupt.
     // Activating ADC0 Common interrupt (Control).
  };
```

EEPROM data is simulated as constant values. I2C communication functions are implemented but just for simulating.

```
void eepromInit(void) {
    printf("EEPROM I2C Communication Start\n");
};

void i2c_write(uint8_t data) {
    printf("I2C Write: 0x%X\n", data);
    // Simulate EEPROM write to an array in memory
  }

void i2c_read(uint8_t byte) {
    printf("I2C Read (Byte: %d)\n", byte);
    // Simulate EEPROM read from the memory array
    if(byte == 0)
        eeprom.sensorType=0x02;
    if(byte == 1);
    eeprom.serialNumber = 0xABC1234;
};

void eepromRead(void){
    i2c_write(0xA1); // EEPROM address (read mode)
    i2c_read(0);
    i2c_read(1);
};
```

LED set and reset features are also implemented as simulating functions.

```
void ledInit(){
    printf("Mock LED initialized on");
    led.Green= false;
    led.Yellow = false;
    led.Red = false;
};

void ledSetGreen() {
    led.Green = true;
    printf("Mock Green LED on ");
};

void ledResetGreen() {
    led.Green = false;
    printf("Mock Green LED off ");
};
```

In main loop , functions are simulated in a specific range. Also ADC readings are done with a specific frequency. In addition, log files are created for visualing simulation results.

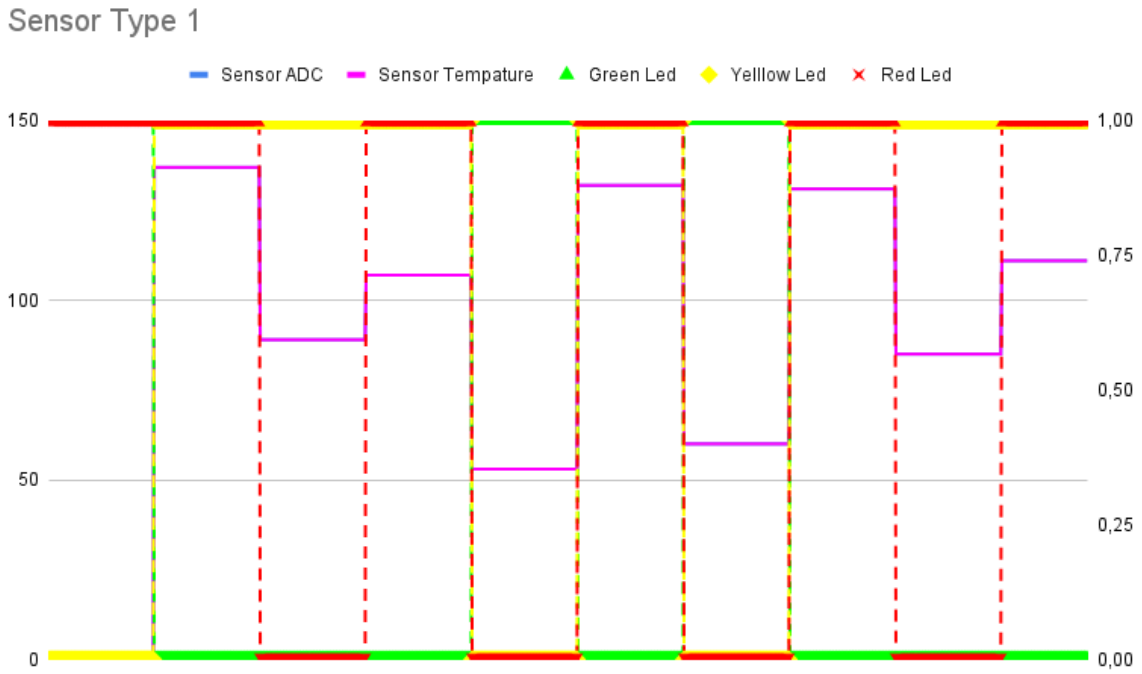
```
while(counterTime<1000)
{
        if (measurementCounter>TEMP_MEASUREMENT_INTERVAL)
        {
            adcRead();
            measurementCounter=0;
        }
        else
            measurementCounter++;

        eepromRead();
        updateTemperature();
        setStatusLed();
        counterTime++;

        fprintf(log_file, " sensor type: %d sensor ADC: %d sensor Temp: %f Green led Status: %s
            Yellow led Status: %s Red led Status: %s\n",
            eeprom.sensorType, adc.sensorReading, adc.sensorTemperature,
            led.Green ? "true" : "false" , led.Yellow ? "true" : "false" , led.Red ? "true" : "false");

}
```

Source can be found in following github link : *https://github.com/zaanem/embedded_task.git*

Gcc compiler is used for compiling. Command for Windows environment can be found below.

gcc src/main.c src/adc.c src/eeprom.c src/led.c -o embeded_task.exe

System Output Plots - Sensor Type 1



System Output Plots - Sensor Type 2