Weather Station with Real-Time Data Visualization

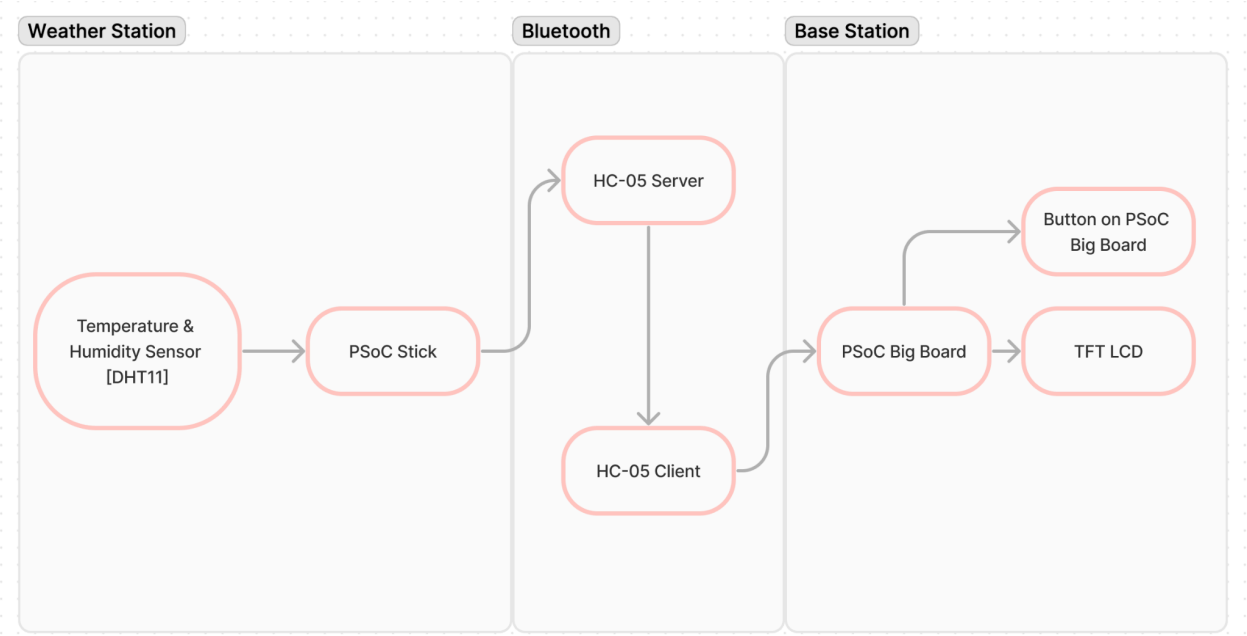
Linh Nguyen

Introduction

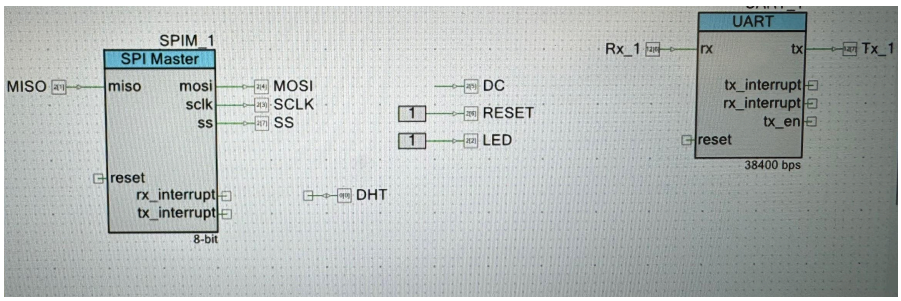
Weather stations play a significant role in climate change research, monitoring, and mitigation efforts. Data collected on temperature, humidity, and pressure is invaluable for climate scientists studying long-term climate trends and predicting future climate scenarios. The objective of this project is to develop a portable weather station that collects and analyzes weather data, which can then be transmitted to a base station for real-time weather data visualization. Both the weather station and the base station will utilize the Programmable System-on-Chip (PSoC) platform as their main architecture. This platform will facilitate the processing of data collected from digital sensors and interpretation of the data to be displayed on a TFT LCD display.

The final product consists of a Weather Station [PSoC Stick] that collects temperature and humidity data through the DHT11 sensor that communicates wirelessly to the Base Station [PSoC Big Board] through the HC-05 Bluetooth Module. The Base Station is connected to a TFT display that displays both the raw temperature and humidity readings from the Weather Station and the graph of how the weather data has changed overtime.

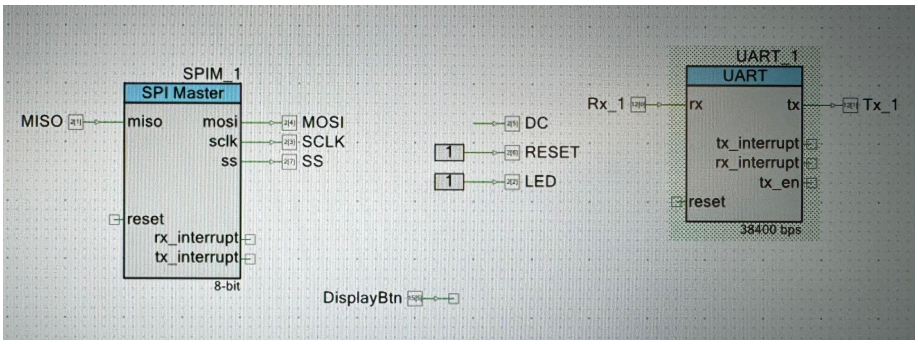
Schematic


Weather Station
Bluetooth
Base Station
Temperature & Humidity Sensor [DHT11]
PSoC Stick
HC-05 Server
HC-05 Client
PSoC Big Board
Button on PSoC Big Board
TFT LCD

Weather Station



Base Station



Temperature and Humidity Sensor [DHT11]

Introduction of DHT11

In the original proposal, the BME280 was the main sensor for obtaining readings for temperature, humidity, and pressure; however, it was difficult to read values and I supplied the sensor more voltage than it could handle, so I switched to a DHT11 sensor instead, which collects values for temperature and humidity.

The Temperature and Humidity readings of the Weather Station use the DHT11 sensor complex, which contains a digital signal output. It incorporates a capacitive humidity sensor and a thermistor to measure the surrounding air, providing digital signal output on the data pin and requiring no analog input pins. The sensor is reliable for humidity readings between 20-80% with an accuracy of 5%, while its temperature readings are accurate within 0-50°C with a deviation of ±2°C. Although the DHT11 is straightforward to use, it necessitates precise timing for data acquisition. Typical applications of the DHT11 are often seen with Arduinos, which have built-in libraries for data parsing. Integrating this sensor with the PSoC platform would involve writing specific code to handle the sensor's digital output and timing requirements.

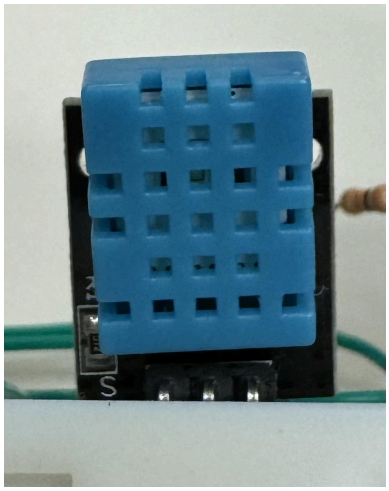
DHT11 uses only one wire [Data] for communication. The voltage level with a certain time value defines the logic one or logic zero on this pin. The communication process is divided in three steps, first is to send a request to the DHT11 sensor then the sensor will send a response pulse and then it starts sending data of total 40 bits to the microcontroller.

Communicating Between PSoC Stick [Weather Station] and DHT11

In the Top Design, the Data pin is set to be a bidirectional pin in order to read the sensor values and write the pulses to communicate with the DHT11. In order to communicate between the DHT11 and the PSoC Stick, the PSoC Stick sends a start pulse to the DHT11 that pulls the Data pin to low for a minimum of 18ms and then pulls up. In order to compensate for timing issues, the PSoC sends a start pulse of 20ms instead to ensure that the Data recognizes that communication has been initiated. After getting the start pulse from the PSoC, the DHT11 sensor sends the response pulse which indicates that

DHT11 received the start pulse. The response pulse is low for 54us and then goes high for 80us. The PSoC counts for these pulses in order to begin collecting data from the DHT11. After sending the response pulse, the DHT11 sensor sends the data, which contains humidity and temperature value along with checksum. The data frame is of total 40 bits long, it contains 5 segments (byte) and each segment is 8-bit long.

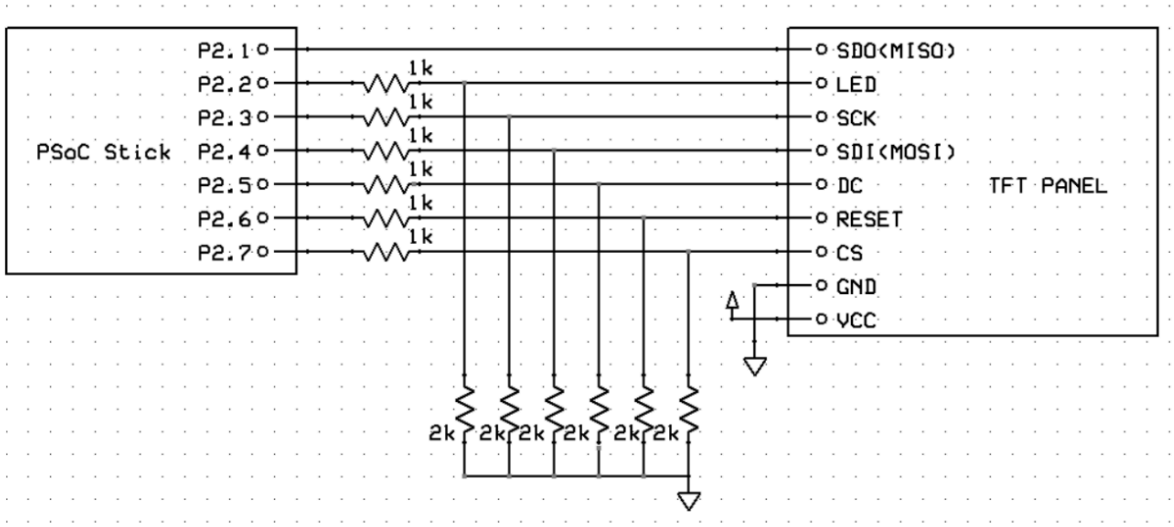
In these 5 segments, the first two segments represent humidity value in decimal integer form, giving Relative Percentage Humidity. The first 8 bits are for the integer part, and the next 8 bits are for the fractional part. The next two segments contain temperature value in decimal integer form, providing temperature in Celsius. The last segment is the checksum, holding the checksum of the first four segments. For the purpose of this project, only the integer value for the temperature and humidity reading is used. Another important component of the DHT11 is that data is obtained every 2 seconds, so between sending pulses to the DHT11 to receive new data, a 2 second delay must be made in order to ensure accurate readings.



TFT Display

Setting Up TFT Display

In order to read the data that is read from the DHT11 sensor, the temperature and humidity readings were stored and displayed on a TFT Display. The TFT was wired on a breadboard using the schematic that Professor Leeb sent out. The reasoning behind wiring the TFT on the separate breadboard was to facilitate easier changes for switching the TFT display between the PSoC stick and the PSoC big board. The TFT also acted as a debugging tool, since the temperature and humidity values were not displayed locally on a terminal. In terms of displaying the values, the EmWin Graphic Library was used.



Bluetooth Module [HC-05]

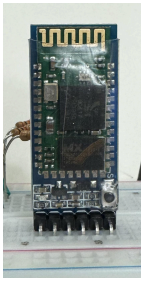
After the data collected on the DHT11 was verified when it displayed on the TFT screen, the HC-05 Bluetooth Module was incorporated to connect the PSoC Stick [Weather Station] to the PSoC Big Board [Base Station]. The original proposal called for the HC-06 Bluetooth Module; however, the module cannot be a host, so the module was switched to the HC-05 instead. The bluetooth HC-05 module was set up through the RS232 bridge of the PSoC Big Board and communication with the HC-05 was done through typing to Tera Term through COM port connection on the Big Board. The HC-05 module had 5 pins: VCC, GND, RX, TX, and EN.

Setting Up HC-05 Host and Receiver

In order to set up the HC-05 to be a host host, the EN had to be set to HIGH in order to put the HC-05 in AT mode. In order for the HC-05 Host and the HC-05 Receiver to communicate to each other, both were set up to have a Baud rate of 38400. Then, through following the AT commands for pairing two HC-05 modules, they were configured and paired together when both are supplied with power. In orderThe pairing between the two modules was confirmed because sending a message on the Tera Term Host HC-05 will send the message on the Receiver HC-05 Module.

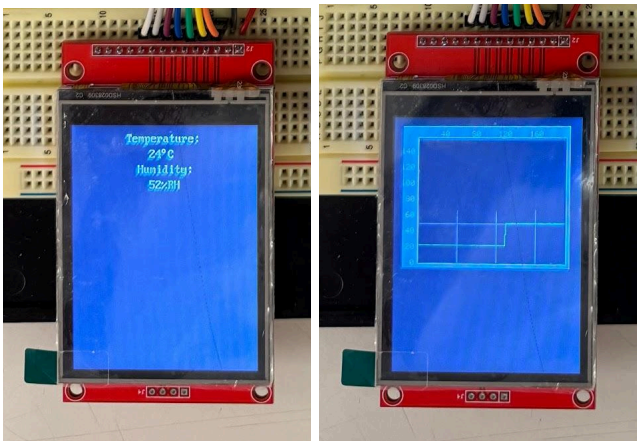
Sending Data from HC-05 Host to Receiver

After the HC-05 modules have been paired with each other, the EN pin was removed from each to ensure they can communicate to each other. The Host HC-05 was wired to the PSoC stick; however, the PSoC stick runs on 5V while the RX and TX lines of the HC-05 module uses 3.3V logic, so a voltage divider was used to ensure 3.3V goes to the RX and TX lines of the HC-05 module. The data was sent through UART communication between the two PSoCs.



Button to Change Display

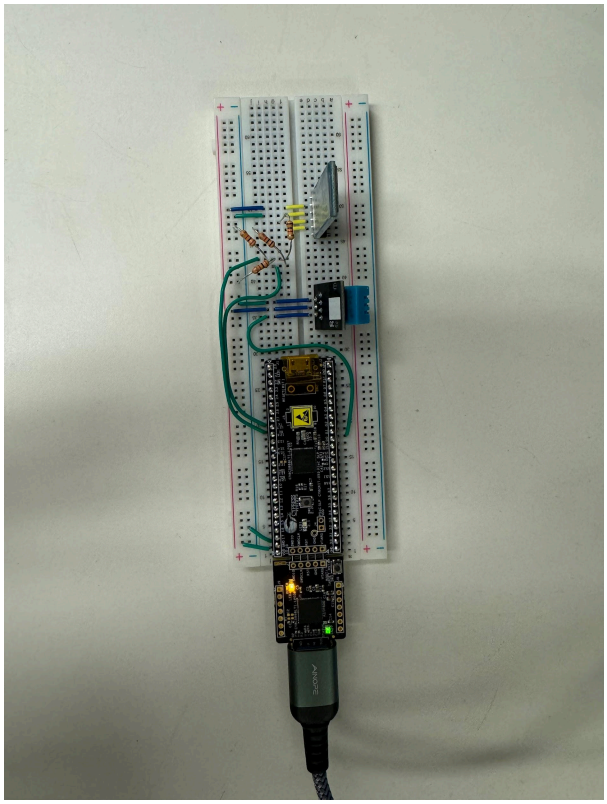
The last component of the Base Station is to change the display of the data with a button. Instead of adding additional buttons to the Base Station [PSoC Big Board], the preexisting buttons of the PSoC Big Board are used to change the display of the TFT screen. The PSoC big board uses a state machine to change between different display states and waits for a signal from the button to change the state of the display. In addition to the display that showcases the numerical values of temperature and humidity, a graph display was added to showcase how temperature and humidity changes over time. The graph display was created using the GRAPH library and the WM library from EmWin.

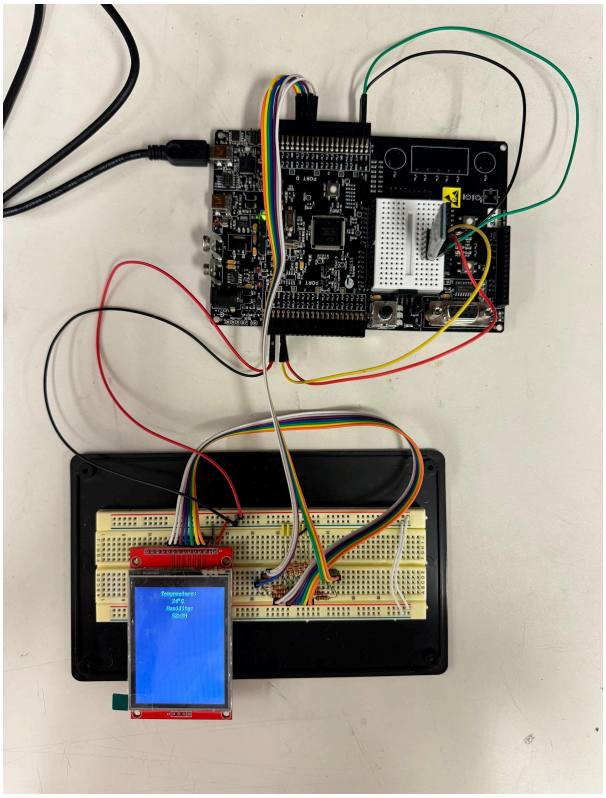


Final Product!!

Weather Station

Base Station





Code

WEATHER STATION

```
/* ========================================
 *
 * Copyright YOUR COMPANY, THE YEAR
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION
 * WHICH IS THE PROPERTY OF your company.
 *
 * ========================================
*/
#include <project.h>
```

```
#include "GUI.h"
#include "tft.h"
#include "stdio.h"
#include "stdlib.h"

void MainTask(void);

int DHTread();

static int temperature=99;
static int humidity=99;
uint8 bits[5];
int DHTread()
{
  int i;
  uint8 Istate;
  Istate=CyEnterCriticalSection();
  //uint8 bits[5];
  uint8 cnt = 7;
uint8 idx = 0;
  int   calc=0;
  int   timeout=0;
  for (i=0; i< 5; i++)
     bits[i] = 0;
  DHT_Write(0u);
  CyDelay(19);
  DHT_Write(1u);
  while(DHT_Read()==1)
  {
     timeout++;
     if(timeout>500)
        goto r99;  //DHT error function
  }
  while(DHT_Read()==0)
  {
     timeout++;
     if(timeout>500)
        goto r99; //DHT error function
  }
  calc=timeout;
  timeout=0;
  while(DHT_Read()==1);
  for (i=0; i<40; i++)
        {
```

```
        timeout=0;
        while(DHT_Read()==0);
        while(DHT_Read()==1)
            timeout++;
        //Data acquiring point
        if ((timeout) > (calc/2))
            bits[idx] |= (1 << cnt);
        if (cnt == 0)   // check for next byte
            {
                    cnt = 7;    // restart at MSB
                    idx++;      // next byte!
            }
            else cnt--;
    }
    humidity    = bits[0];
    temperature = bits[2];
    CyExitCriticalSection(Istate);
    CyDelay(1);
    return 0;
    r99:    //Goto label for error in DHT reading
        humidity    = 99;
        temperature = 99;
        CyExitCriticalSection(Istate);
        return 99;
}

int main()
{
    CyGlobalIntEnable;                  // Enable global interrupts
    SPIM_1_Start();                     // initialize SPIM component
    UART_1_Start();
    GUI_Init();                         // initilize graphics library
    GUI_Clear();
    GUI_SetFont(&GUI_Font8x16);
    for(;;) {
        DHTread();
        int i;
        //write temp and humidity data to Base station
        for (i=0; i < 2; i++ ) {
            if(UART_1_ReadTxStatus() & UART_1_TX_STS_FIFO_NOT_FULL) {
                uint8 sensorOutput[30] = {temperature, humidity};
                UART_1_WriteTxData(sensorOutput[i]);
                CyDelay(10);
            }
```

```
        }

        CyDelay(2000);  //Delay in milli seconds
    }
}

/* [] END OF FILE */
```

BASE STATION

```
/* ========================================
 *
 * Copyright YOUR COMPANY, THE YEAR
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION
 * WHICH IS THE PROPERTY OF your company.
 *
 * ========================================
*/
#include <project.h>
```

```
#include "GUI.h"
#include "tft.h"
#include "stdio.h"
#include "stdlib.h"
#include "GRAPH.h"
#include "WM.h"


uint8 temp = 0;
uint8 humidity = 0;
int displayState = 0;

void numDisplay() {
   //get humidity value from weather station
   if(UART_1_ReadRxStatus() & UART_1_RX_STS_FIFO_NOTEMPTY) {
      char humidityStr [40]; //init humidity string
      UART_1_ClearRxBuffer();
      humidity = UART_1_ReadRxData();
      GUI_DispStringAt("Humidity:", 85, 50);
      sprintf(humidityStr, "%i%%RH", humidity);
      GUI_DispStringAt(humidityStr,100,70); //display humidity on TFT
   }
   CyDelay(10);
   //get temp value from weather station
   if(UART_1_ReadRxStatus() & UART_1_RX_STS_FIFO_NOTEMPTY) {
      char tempStr [40];
      UART_1_ClearRxBuffer(); //clear buffer get new data value
      temp = UART_1_ReadRxData(); //get temp
      GUI_DispStringAt("Temperature:", 70, 10);
      sprintf(tempStr, "%i%cC", temp, 0xB0);
      GUI_DispStringAt(tempStr,100,30); //display temp on TFT
      //GUI_DispNextLine();
   }
}

short tempData[200];
short humidityData[200];

GRAPH_DATA_Handle hData;
GRAPH_SCALE_Handle hScale;
GRAPH_SCALE_Handle vScale;
WM_HWIN hGraph;
```

```
void graphTempDisplay(){
   //init arbituary data to make graph
   const short data[23] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
   WM_MULTIBUF_Enable(1);
   hGraph = GRAPH_CreateEx(10,10,216,180,WM_HBKWIN,WM_CF_SHOW,
GRAPH_CF_GRID_FIXED_X,GUI_ID_GRAPH0);
   //init graph info
   GRAPH_SetGridDistX(hGraph,50);
   GRAPH_SetGridDistY(hGraph,50);
   GRAPH_SetVSizeX(hGraph,50);
   GRAPH_SetGridVis(hGraph,1);
   GRAPH_SetBorder(hGraph,20,15,5,5);
   //create graph Object w appropriate scale
   hScale = GRAPH_SCALE_Create(16, GUI_TA_RIGHT, GRAPH_SCALE_CF_VERTICAL,
20);
   GRAPH_AttachScale(hGraph, hScale);
   vScale = GRAPH_SCALE_Create(8, GUI_TA_RIGHT,
GRAPH_SCALE_CF_HORIZONTAL,40);
   GRAPH_AttachScale(hGraph, vScale);
   //attach data to graph
   hData = GRAPH_DATA_YT_Create(GUI_DARKGREEN, 200,data, 23);
   GRAPH_DATA_YT_SetAlign(hData, GRAPH_ALIGN_LEFT);
   GRAPH_AttachData(hGraph, hData);

   GUI_Delay(10u);
   //collect temp values to display on TFT screen
   int i;
   for (i = 0; i < 200; i++) {
      //get humidity value to store
      if(UART_1_ReadRxStatus() & UART_1_RX_STS_FIFO_NOTEMPTY) {
      UART_1_ClearRxBuffer();
      humidity = UART_1_ReadRxData();
      }
      CyDelay(10);
      if(UART_1_ReadRxStatus() & UART_1_RX_STS_FIFO_NOTEMPTY) {
      UART_1_ClearRxBuffer();
      temp = UART_1_ReadRxData();
   }
      //collect only temp values to display
      tempData[i] = temp;
   }
   GRAPH_DATA_YT_Delete(hData);
   hData = GRAPH_DATA_YT_Create(GUI_DARKGREEN,220,tempData,200);
   GRAPH_DATA_YT_SetAlign(hData, GRAPH_ALIGN_LEFT);
```

```
    GRAPH_AttachData(hGraph,hData);
    GUI_Delay(10u);
}

void graphHumidityDisplay(){
    GUI_DispString("graph humidity");
}




int main()
{
    CyGlobalIntEnable;                  // Enable global interrupts
    SPIM_1_Start();                     // initialize SPIM component
    UART_1_Init();
    UART_1_Start();
    UART_1_ClearRxBuffer();
    GUI_Init();
    GUI_Clear();
    GUI_SetFont(&GUI_Font8x16);
    int displayState = 0;
    //go through state machine to display temp and humidity values
    for(;;) {
        if(!DisplayBtn_Read()) {
            GUI_Clear();
            WM_DeleteWindow(hGraph);
            while (!DisplayBtn_Read()){
            }
            if(displayState==2) {
                displayState = 0;
            } else {
                displayState += 1;
            }
        }

        if (displayState == 0) {
            numDisplay();
        } else if (displayState == 1) {
            graphTempDisplay();
        } else if (displayState == 2) {
            graphHumidityDisplay();
        }
    }
```

```
}

/* [] END OF FILE */
```