

//Written By: Current Solutions ECEE Capstone 25/26
//Ryan Buckley, Connor Sorrell, Faris Balubaid, South Livermoore, Ilya Vitlviskiy
//Last Update 04/22/2026
//Questions? Email: rbuckley2003@gmail.com

// =====
// High level overview:
// We have a loop that reads all sensors then prints these values to the serial monitor
// The serial monitor on the Arduino correlates to the tx pin (transmit) this pin then
// Writes the data to the open log (SD card), the data is seperated by commas so when
// inserted into excel columns can be easily seperated
// =====

//Necesary libraries for functionality
//Each library is typicall a sensor
//If you are curious as to how these libraries work you can view their documentation online
#include <Wire.h>
#include <Adafruit_LIS3MDL.h>
#include <Adafruit_LSM6DSOX.h>
#include <SensirionI2cScd4x.h>
#include <Adafruit_BME680.h>
#include <Adafruit_INA219.h>
#include <OneWire.h>
#include "Adafruit_MAX31855.h"
#include <DallasTemperature.h>

//Pin definitations
// MAX31855 pins
#define MAXDOEXTTEMP 2
#define MAXCSEXTTEMP 3
#define MAXCLKEXTTEMP 4

// Indicator LEDs
#define BMEINDICATOR 10
#define IMUINDICATOR 11
#define OLINDICATOR 8
#define EXTTEMPINDICATOR 13
#define INAINDICATOR 12

// RGB LED pins
#define VOLTAGE1 5 // Red
#define VOLTAGE2 6 // Green
#define VOLTAGE3 9 // Blue

#define ONE_WIRE_BUS 7

// =====

//Initialize all sensor statuses to false
bool BMEStatus = false;
bool LSMStatus = false;
bool LISStatus = false;
bool INASStatus = false;
bool MAXStatus = false;

```

//Initialize the external temperature probe
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

//Initializing objects utilizing libraries each object corresponds to a sensor
Adafruit_LIS3MDL lis3mdl;
Adafruit_LSM6DSOX lsm6dsox;
SensirionI2cScd4x scd41;
Adafruit_BME680 bme688;
Adafruit_INA219 ina219 = Adafruit_INA219();

//Initalize the external temperature probe
Adafruit_MAX31855 thermocouple(MAXCLKEXTTEMP, MAXCSEXTTEMP, MAXDOEXTTEMP);

//Create constants for the RGB Leds
const int RED_PIN = VOLTAGE1;
const int GREEN_PIN = VOLTAGE2;
const int BLUE_PIN = VOLTAGE3;

//Function for setting the color of the RGB LED
void setColor(int red, int green, int blue) {
    analogWrite(RED_PIN, red);
    analogWrite(GREEN_PIN, green);
    analogWrite(BLUE_PIN, blue);
}

//Function for updating the RGB Led based off the battery voltage
//Note the fs on these values indicate that we are dealing with float literal values
void updateBatteryColor(float batteryV) {
    //If battery greater than 9 volts indicate green
    if (batteryV > 9.0f) {
        setColor(0, 100, 0);    // Green
        //If battery voltage is greater than or equal to 8 volts set the RGB LED to yellow
    } else if (batteryV >= 8.0f) {
        setColor(80, 20, 0);    // Yellow
    } else {
        setColor(100, 0, 0);    // Red
    }
}

// =====

//Setup the entire system
void setup() {
    //Initialize our baud rate to 9600
    //IMPORTANT keep baud rate low in order to prevent corruption on the SD card
    Serial.begin(9600);
    delay(3000);

    //Wait for serial to begin before proceeding
    while (!Serial) { ; }

    Wire.begin();
    //3 Second delay to allow the system to setup
    delay(3000);

```

```
//Begin all of our sensors
sensors.begin();
```

```
//Clarify the function of each pin being used
pinMode(BMEINDICATOR, OUTPUT);
pinMode(IMUINDICATOR, OUTPUT);
pinMode(OLINDICATOR, OUTPUT);
pinMode(EXTTEMPINDICATOR, OUTPUT);
pinMode(INAINDICATOR, OUTPUT);
```

```
pinMode(RED_PIN, OUTPUT);
pinMode(GREEN_PIN, OUTPUT);
pinMode(BLUE_PIN, OUTPUT);
```

```
//Start with all indicator LEDs off
digitalWrite(BMEINDICATOR, LOW);
digitalWrite(IMUINDICATOR, LOW);
digitalWrite(OLINDICATOR, LOW);
digitalWrite(EXTTEMPINDICATOR, LOW);
//Start with the RGB LED turned off
setColor(0, 0, 0);
```

```
//Create file FLIGHTDATA.txt
//This is the text file that all data logged will be written to
Serial.print("new FLIGHTDATA.TXT\r");
```

```
Serial.print(char(26));
Serial.print(char(26));
Serial.print(char(26));
//If we have a flightdata.txt append to that file
Serial.print("append FLIGHTDATA.TXT\r");
Serial.println();
```

```
//Sensor initialization if we do not find the specific the specific sensor we are looking for print a
message
```

```
//If we do find the sensor we are looking for turn on the indicator LED
```

```
if (!IIS3MDL.begin_I2C(0x1C)) {
    Serial.println(" LIS3MDL not found!");
}
else {
    Serial.println("LIS3MDL Initialized");
    LISStatus = true;
}
```

```
if (!LSM6DSOX.begin_I2C(0x6A)) {
    Serial.println("LSM6DSOX not found!");
}
else {
    Serial.println("LSM Initialized");
    LSMStatus = true;
}
```

```
// IMU indicator lights when BOTH are good
if (LISStatus && LSMStatus) {
```

```

    digitalWrite(IMUINDICATOR, HIGH);
}

if (!bme688.begin(0x76)) {
    Serial.println("BME688 not found!");
}
else {
    Serial.println("BME Initialized");
    digitalWrite(BMEINDICATOR, HIGH);
    BMESStatus = true;
}

if (!ina219.begin()) {
    Serial.println("INA219 not found!");
}
else {
    Serial.println("INA Initialized");
    digitalWrite(INAINDICATOR, HIGH);
    INASStatus = true;
}

//Get both the internal and external temperature readings from our probe
double tInt = thermocouple.readInternal();
double tExt = thermocouple.readCelsius();
//If we get a proper number we know that our sensor is working
//If we do not get a number we know that the max is not working and will send a message
if (!isnan(tInt) && !isnan(tExt)) {
    MAXStatus = true;
    digitalWrite(EXTTEMPINDICATOR, HIGH);
    Serial.println("MAX31855 Initialized");
} else {
    Serial.println("MAX31855 read failed!");
}

delay(2000);

//Intro statement will make the header of our file this indicates what each value being printed out
represents
introStatment();
delay(2000);

Serial.print(", ");
//Sync is the openlogs version of flush, saving the values written to our SD card
Serial.print("sync\r");
}

void loop()
{
    //Indicator for the start of our loop
    digitalWrite(OLINDICATOR, HIGH);
    //Function for printing our time to serial monitor
    getTimeStamp();
    //Function for printing the values from the BME to serial monitor
    getBMEReading();
    //Function for printing the values from the LIS3MDL to serial monitor
    getLIS3MDLding();
}

```

```
//Function for printing the values from the LSM6DSOX to serial monitor  
getLSM6DSOXReading();
```

```
//Obtaining the voltage from the INA sensor and using that value to indicate on our LED what voltage  
we are at
```

```
float loadV = getINAResult();  
if (INAStatus && !isnan(loadV)) {  
    updateBatteryColor(loadV);  
}
```

```
//Function for printing the values from the external temperature probe to the serial monitor  
getProbeReading();
```

```
Serial.println("sync\r"); //Openlogs version of flush saves the data sequentially to prevent lost data
```

```
//Indicator that we have properly gone through the loop  
digitalWrite(OLINDICATOR, LOW);
```

```
}
```

```
//Function for printing our names of values to serial monitor
```

```
//When data is eventually added to excel these will be the headers of the columns  
void introStatment()
```

```
{  
    Serial.println("Initialization complete");  
    Serial.print("Time, ");  
    delay(10);  
    Serial.print("Temp C, ");  
    delay(10);  
    Serial.print("Humid, ");  
    delay(10);  
    Serial.print("Altitude M, ");  
    delay(10);  
    Serial.print("Gas Resis, ");  
    delay(10);  
    Serial.print("Magx, ");  
    delay(10);  
    Serial.print("Magy, ");  
    delay(10);  
    Serial.print("Magz, ");  
    delay(10);  
    Serial.print("Accel X, ");  
    delay(10);  
    Serial.print("AccelY, ");  
    delay(10);  
    Serial.print("AccelZ, ");  
    delay(10);  
    Serial.print("Gyrox, ");  
    delay(10);  
    Serial.print("Gyroy, ");  
    delay(10);  
    Serial.print("GyroZ, ");  
    delay(10);  
    Serial.print("Bus, ");  
    delay(10);  
    Serial.print("Shunt, ");  
    delay(10);  
}
```

```

    Serial.print("Current, ");
    delay(10);
    Serial.print("Load, ");
    delay(10);
    Serial.print("Temp I, ");
    delay(10);
    Serial.println("Temp E");
}

//Function for getting our current time
void getTimeStamp()
{
    float timeStamp = 0;
    timeStamp = (millis() / 1000.0);
    Serial.print(timeStamp);
    Serial.print(", ");
}

//Function for getting reading from the BME
void getBMEReading()
{
    //Only perform reading if the BME is working
    if (bme688.performReading() && BMEStatus == true)
    {
        //Temperature reading from BME (internal)
        Serial.print(bme688.temperature);
        Serial.print(", ");

        //Humidity reading from the BME
        Serial.print(bme688.humidity);
        Serial.print(", ");

        //Grabbing pressure value from the BME then using this value to calculate what our altitude is
        float pressure_hPa = (bme688.pressure / 100.0);
        float altitude_m = 44330.0 * (1.0 - pow(pressure_hPa / 1013.25, 0.1903));
        Serial.print(altitude_m);
        Serial.print(", ");

        //Print gas resistance to serial monitor
        Serial.print(bme688.gas_resistance / 1000.0);
        Serial.print(", ");

        delay(25);
    }
    else
    {
        //If we cannot get our values from the BME indicate that we have a failure for debugging purposes
        Serial.println("BME FAILURE");
    }
}

//Function for getting our magnetic readings to the SD card
void getLIS3MDLding()
{
    sensors_event_t mag;
    if (lis3mdl.getEvent(&mag))

```

```

{
    //X magnetic reading
    Serial.print(mag.magnetic.x);
    Serial.print(", ");
    //Y magnetic reading
    Serial.print(mag.magnetic.y);
    Serial.print(", ");
    //Z magnetic readin
    Serial.print(mag.magnetic.z);
    Serial.print(", ");
}
else
{
    //In the case of a failure print to serial montior so we can debug the issue
    Serial.println("LIS3MDL FAILURE");
}
}
//Function for getting our 9 degree IMU readings to the SD card
void getLSM6DSOXReading()
{
    if (LSMStatus == true)
    {
        sensors_event_t accel, gyro, temp2;
        lsm6dsox.getEvent(&accel, &gyro, &temp2);
        //Acceleration on the X axis
        Serial.print(accel.acceleration.x);
        Serial.print(", ");
        //Acceleration on the Y axis
        Serial.print(accel.acceleration.y);
        Serial.print(", ");
        //Acceleration on the Z axi
        Serial.print(accel.acceleration.z);
        Serial.print(", ");

        //Gyroscope reading on the x axis
        Serial.print(gyro.gyro.x);
        Serial.print(", ");
        //Gyroscope reading on the y axis
        Serial.print(gyro.gyro.y);
        Serial.print(", ");
        //Gyroscope reading on the z axis
        Serial.print(gyro.gyro.z);
        Serial.print(", ");
    }
}

//Function for printing our INA values to the SD card
float getINAReading()
{
    //Initializing load voltage as not a number as we will calculate this value
    float loadVoltage = NAN;

    //If the INA is working
    if (INAStatus == true)
    {
        //Get shunt voltage and bus voltage

```

```

float shuntVoltage = ina219.getShuntVoltage_mV(); // millivolts
float busVoltage = ina219.getBusVoltage_V(); // volts
//Calculate load voltage
loadVoltage = busVoltage + (shuntVoltage / 1000.0);

//Printing out the bus voltage
Serial.print(busVoltage);
Serial.print(", ");

//Print out the shunt voltage
Serial.print(shuntVoltage);
Serial.print(", ");

//Current value (in mA)
float current = ina219.getCurrent_mA();
Serial.print(current);
Serial.print(", ");

//Print out the load voltage
Serial.print(loadVoltage);
Serial.print(", ");
}
//Return the load voltage value so that we can use it in the case of setting the color of RGB LED
return loadVoltage;
}

//Function for getting the probe reading
void getProbeReading()
{
    //Gettign and printing the internal and external temperature
    double externalTemp = thermocouple.readCelsius();
    double internalTemp = thermocouple.readInternal();

    Serial.print(internalTemp);
    Serial.print(", ");
    Serial.print(externalTemp);
    Serial.print(", ");
}

//Unused function but can be used to blink an LED pass in the pin number of the LED you want to blink
void blinkLED(int LEDTOBLINK)
{
    digitalWrite(LEDTOBLINK, HIGH);
    delay(250);
    digitalWrite(LEDTOBLINK, LOW);
}

```