```
// Last edit 3/23/2022

#include "mbed.h"

#include <iostream>


AnalogIn LightSensor(PTB0);

AnalogIn TemperatureSensor(PTB1);

AnalogIn TorqueSensor(PTB2);

InterruptIn START_BUTTON(PTD4);

InterruptIn STOP_BUTTON(PTA12);

DigitalOut RED_LED(LED1);

DigitalOut GREEN_LED(LED2);

DigitalOut BLUE_LED(LED3);

DigitalOut OUTPUT(PTC2);


#define Vsupply 3.3f //microcontroller voltage supply 3.3V vin


//variables for light sensor

float LightSensorDigiValue; //the A/D converter value read by the controller input pin

float LightSensorVoltValue; //voltage on the controller input pin

float LdrResistance; //photoresistance value

#define LdrBiasResistor 3300.0f //Bias resistor (upper leg of voltage divider) for LDR


//variables for temperature sensor

float TemperatureSensorDigiValue; //the A/D converter value read by the controller input pin

float TemperatureSensorVoltValue; //the voltage on the controller input pin (across the 10k resistor) from the temperature sensor voltage
divider vo

float ThermistorResistance; //computed from the voltage drop value across the thermistor

float ThermistorTemperature; //approximate ambient temperature measured by thermistor

#define ThermistorBiasResistor 10000.0f //Bias resistor (lower leg of voltage divider) for thermistor r2


//variables for torque sensor

float MotorCurrentDigiValue; //the A/D converter value ready by the controller input pin

float MotorCurrentVoltValue; //the voltage on the controller input pin (across the 10 ohm resistor) from the motor torque sensor

float MotorCurrent; //computed from the voltage value

#define MotorSeriesResistance 10.0f //resistance of torque (current) sensing resistor in series with the Motor


// Variables to hold control reference values.

// STUDENT: EDIT THESE VALUES

float LightResistanceLimit = 8000.0; //enter a resistance reference for LDR load activation

float TemperatureLimit = 27.0; //enter a temperature in Celsius here for temperature deactivation; NOTE: room temperature is 25C

float MotorCurrentLimit = 0.1; //enter a reference current in amperes for motor torque deactivation


// This function will be attached to the start button interrupt.

void StartPressed(void)
```

```
{

    cout << "Start!" << endl;

    OUTPUT = 1;

}

// This function will be attached to the stop button interrupt.
void StopPressed(void)
{
    // STUDENT: EDIT HERE

    cout << "Stop!" << endl;

    OUTPUT = 0;

}

//convert the input voltage from the light sensor to an LDR resistance value
//Resistance is inversely proportional to the amount of light

float getPhotoResistance(void)
{
    LightSensorDigiValue = LightSensor.read(); //read the LightSensor A/D value
    LightSensorVoltValue = Vsupply*LightSensorDigiValue; //convert to voltage
    LdrResistance = LightSensorVoltValue*LdrBiasResistor/(Vsupply - LightSensorVoltValue); //voltage divider equation to determine LDR
resistance

    return LdrResistance;
}

// This function will check the LDR analog input.
// STUDENT: USE THIS AS AN EXAMPLE FOR THE TEMPERATURE AND TORQUE CHECK FUNCTIONS
void CheckLightSensor(void)
{
    if(getPhotoResistance() >= LightResistanceLimit) {
        cout << "LDR Start!" << endl;
        OUTPUT = 1;
        GREEN_LED = 0; // ON
    }
    else {
        GREEN_LED = 1; // OFF
    }
}

// This function converts the voltage value from the thermistor input to an approximate temperature
// in Celsius based on a linear approximation of the thermistor.
float getThermistorTemperature(void)
{
    // STUDENT: EDIT HERE
    // 1. Read the TemperatureSensor A/D value and store it in TemperatureSensorDigiValue
     TemperatureSensorDigiValue = TemperatureSensor.read();
    TemperatureSensorVoltValue = Vsupply*TemperatureSensorDigiValue; //convert to voltage
    // 2. Calculate TemperatureSensorVoltValue from TemperatureSensorDigiValue and Vsupply
    // 3. Calculate ThermistorResistance using the voltage divider equation
    ThermistorResistance=((Vsupply*ThermistorBiasResistor)/TemperatureSensorVoltValue)- ThermistorBiasResistor;

    ThermistorTemperature = ((ThermistorResistance - 10000.0)/(-320.0)) + 25.0; //temperature of the thermistor computed by a linear
approximation of the device response

    return ThermistorTemperature;
}

//This function will check for a temperature triggered deactivation of the motor
void CheckTemperatureSensor(void)
{
    // STUDENT: EDIT HERE
    // Use the getThermistorTemperature() function defined above to obtain a temperature value to use for comparison and decision making with
your TemperatureLimit


    if(getThermistorTemperature() >= TemperatureLimit) {
        cout << " thermistor stop!" << endl;
        OUTPUT = 0;
        RED_LED = 0; // ON
    }
    else {
        RED_LED = 1; // OFF
    }

}

//This function will determine the motor current in amperes
float getMotorCurrent(void)
{
```

```
    // STUDENT: EDIT HERE
    // 1. Read the TorqueSensor value and store it in MotorCurrentDigiValue
     MotorCurrentDigiValue = TorqueSensor.read();
    // 2. Calculate MotorCurrentVoltValue from MotorCurrentDigiValue and Vsupply
    MotorCurrentVoltValue = Vsupply*MotorCurrentDigiValue;
    // 3. Calculate MotorCurrent using Ohm's law from MotorCurrentVoltValue and MotorSeriesResistance
    MotorCurrent = MotorCurrentVoltValue/MotorSeriesResistance;
    return MotorCurrent;
}

// This function will check the Over Torque analog input.
void CheckTorqueSensor(void)
{
    // STUDENT: EDIT HERE
    // Use the getMotorCurrentValue() function defined above to obtain a current torque value to use for comparison and decision making with
your MotorCurrentLimit

    if(getMotorCurrent() >= MotorCurrentLimit) {
        cout << "  Motor stop!" << endl;
        OUTPUT = 0;
        BLUE_LED = 0; // ON
    }
    else {
        BLUE_LED = 1; // OFF
    }

}

// Standard entry point in C++.
int main(void)
{
    // Attach the functions to the hardware interrupt pins.
    START_BUTTON.rise(&StartPressed);
    STOP_BUTTON.rise(&StopPressed);
    // Initialize LED outputs to OFF (LED logic is inverted)
    RED_LED = 1;
    GREEN_LED = 1;
    BLUE_LED = 1;

    // Blink the blue LED once to indicate the code is running.
    BLUE_LED = !BLUE_LED;
    wait(1.0);
    BLUE_LED = !BLUE_LED;

    while(true) {
        // Check the analog inputs.
        CheckLightSensor();
        CheckTemperatureSensor();
        CheckTorqueSensor();

        // Print Analog Values to screen
        cout << "\n\rLDR Resistance: " << getPhotoResistance() << endl;
        cout << "\rCurrent Temperature Value: " << getThermistorTemperature() << endl;
        cout << "\rMotor Current: " << getMotorCurrent() << endl;

        wait(1.0); // Wait 1 second before repeating the loop.
    }
}
// End of HardwareInterruptSeedCode
```