To: Christopher Peters
From: Abdullah Khalid

Purpose

The purpose of this project was to create an user interactive sensor monitoring and control system using an Arduino and the GY-87 IMU module, along with an ultrasonic distance sensor and two LEDs. Temperature and motion data was detected, send it to Node-RED through serial communication, and visualized on a live dashboard. The system was designed to allow user control of the red and blue LEDs from the same dashboard, demonstrating two-way communication between hardware and software components.

Methodology / Approach

On the hardware side, the Arduino interfaced with the GY-87 sensor, which gave data on acceleration, gyroscope, temperature, pressure, altitude, and compass heading. An HC-SR04 ultrasonic sensor was used to measure object distance. The output was formatted in JSON and sent over the serial connection. Commands like R,1 or B,128 were used to control the red LED (on/off) and the blue LED's brightness. On the Node-RED side, serial input from the Arduino was parsed using a JSON node and displayed dashboard widgets like gauges and text fields. Control of the LEDs was achieved through a switch and slider widget, each linked to a function node that formatted the output command and sent it back via a serial output node. The blue LED used PWM values (B,0–255), while the red LED used a binary toggle (R,0 or R,1).

Results

The final system worked as intended, successfully transmitting real-time sensor data from the Arduino to the Node-RED dashboard. All values were displayed clearly and updated in real-time. The blue and red LED responded immediately to changes made through the slider. This delay was minimized by isolating the red LED logic into its own function node. No major issues were encountered with sensor accuracy or serial communication.

Appendix

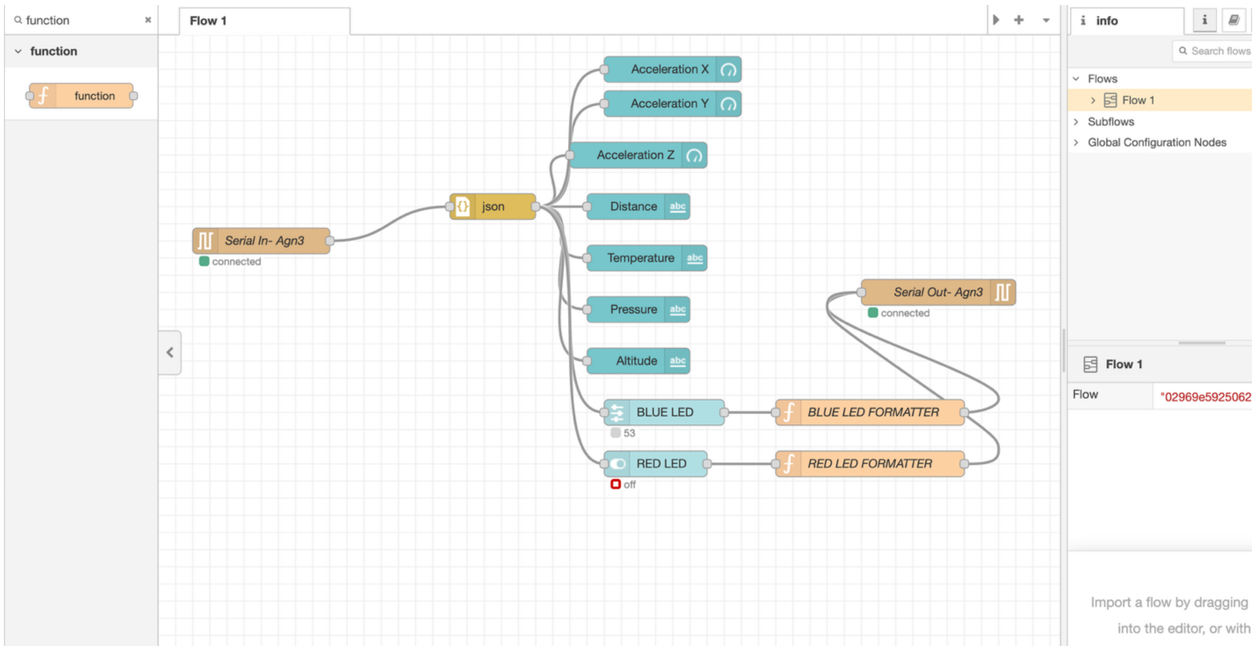
A. Arduino Code:

```
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>
#include <Adafruit_BMP085.h>
#include <QMC5883LCompass.h>

Adafruit_MPU6050 mpu;
Adafruit_BMP085 bmp;
QMC5883LCompass compass;

#define TRIG_PIN 9
#define ECHO_PIN 10
#define RED_LED_PIN 5
#define BLUE_LED_PIN 6

String input = "";

void setup() {
  Serial.begin(9600);
  Serial.println("Serial working...");

  initializeMPU6050();
  mpu.setI2CBypass(true);
  initializeQMC5883L();
  initializeBMP180();

  pinMode(RED_LED_PIN, OUTPUT);
  pinMode(BLUE_LED_PIN, OUTPUT);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  printAllSensors();
}

void loop() {
  handleSerialCommands();
}

void initializeMPU6050() {
  if (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip");
    while (1);
```

```cpp
  }
  Serial.println("MPU6050 Found!");
  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
  mpu.setGyroRange(MPU6050_RANGE_500_DEG);
  mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);
  delay(100);
}

void initializeBMP180() {
  if (!bmp.begin()) {
    Serial.println("Could not find a valid BMP180 sensor, check wiring!");
    while (1);
  }
  Serial.println("BMP180 Found!");
}

void initializeQMC5883L() {
  compass.init();
}

void handleSerialCommands() {
  while (Serial.available()) {
    char c = Serial.read();
    if (c == '\n') {
      input.trim();

      if (input.startsWith("R,")) {
        int state = input.substring(2).toInt();
        digitalWrite(RED_LED_PIN, state ? HIGH : LOW);
        Serial.print("Set RED LED: "); Serial.println(state);
      }
      else if (input.startsWith("B,")) {
        int brightness = input.substring(2).toInt();
        brightness = constrain(brightness, 0, 255);
        analogWrite(BLUE_LED_PIN, brightness);
        Serial.print("Set BLUE LED brightness: "); Serial.println(brightness);
      }
      else {
        Serial.print("Invalid command: "); Serial.println(input);
      }

      input = "";
    } else {
      input += c;
    }
  }
}
```

```
float readUltrasonicDistance() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  long duration = pulseIn(ECHO_PIN, HIGH, 25000);
  if (duration == 0) return -1;
  return (duration * 0.0343) / 2.0;
}

void printAllSensors() {
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);
  compass.read();

  int heading = compass.getAzimuth();
  char dir[4];
  compass.getDirection(dir, heading);

  float temperature = bmp.readTemperature();
  int32_t pressure = bmp.readPressure();
  float altitude = bmp.readAltitude();
  float distance = readUltrasonicDistance();

  Serial.println("\n--- SENSOR OUTPUT ---");
  Serial.println("{");
  Serial.print("  \"acceleration\": {\n");
  Serial.print("    \"x\": "); Serial.print(a.acceleration.x); Serial.println(",");
  Serial.print("    \"y\": "); Serial.print(a.acceleration.y); Serial.println(",");
  Serial.print("    \"z\": "); Serial.print(a.acceleration.z); Serial.println();
  Serial.println("  },");
  Serial.print("  \"gyroscope\": {\n");
  Serial.print("    \"x\": "); Serial.print(g.gyro.x); Serial.println(",");
  Serial.print("    \"y\": "); Serial.print(g.gyro.y); Serial.println(",");
  Serial.print("    \"z\": "); Serial.print(g.gyro.z); Serial.println();
  Serial.println("  },");
  Serial.print("  \"temperature\": "); Serial.print(temperature); Serial.println(",");
  Serial.print("  \"pressure\": "); Serial.print(pressure); Serial.println(",");
  Serial.print("  \"altitude\": "); Serial.print(altitude); Serial.println(",");
  Serial.print("  \"heading\": "); Serial.print(heading); Serial.println(",");
  Serial.print("  \"direction\": \"");
  Serial.print(dir[0]);
  if (dir[1] != '\0') Serial.print(dir[1]);
  if (dir[2] != '\0') Serial.print(dir[2]);
  Serial.println("\",");
  Serial.print("  \"distance\": "); Serial.println(distance);
  Serial.println("}");
  Serial.println("----------------------");
}
```

B. Node-RED Flow JSON:



function
function
function
Flow 1
Acceleration X
Acceleration Y
Acceleration Z
json
Distance
Serial In- Agn3
connected
Temperature
Pressure
Serial Out- Agn3
connected
Altitude
BLUE LED
53
BLUE LED FORMATTER
RED LED
off
RED LED FORMATTER
info
Search flows
Flows
Flow 1
Subflows
Global Configuration Nodes
Flow 1
Flow
"02969e5925062
Import a flow by dragging
into the editor, or with