```
#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>
#include <TaskScheduler.h>
#include <ESP32Servo.h>
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include "Adafruit_BMP3XX.h"
#include "SparkFun_BNO08x_Arduino_Library.h"

#define SERVICE_UUID "9a8ca9ef-e43f-4157-9fee-c37a3d7dc12d"
#define BLINK_UUID "e94f85c8-7f57-4dbd-b8d3-2b56e107ed60"
#define SERVO_UUID "f74fb3de-61d1-4f49-bd77-419b61d188da"
#define BMP390_UUID "94cbc7dc-ff62-4958-9665-0ed477877581"
#define BNO08X_UUID "c91b34b8-90f3-4fee-89f7-58c108ab198f"
#define PIN_UUID "21072f3b-b950-4b29-bf97-9a7be82d93e7"

#define UUID_4 "78c611e3-0d36-491f-afe4-60ecc0c26a85"
#define UUID_5 "6492bdaa-60e3-4e8a-a978-c923dec9fc37"
#define UUID_6 "56e48048-19da-4136-a323-d2f3e9cb2a5d"
#define UUID_7 "83e6a4bd-8347-409c-87f3-d8c896f15d3d"
#define UUID_8 "680f38b9-6898-40ea-9098-47e30e97dbb5"
#define UUID_9 "fb02a2fa-2a86-4e95-8110-9ded202af76b"
#define UUID_10 "a979c0ba-a2be-45e5-9d7b-079b06e06096"

#define DEVICE_NAME "ESP_32"
```

```
#define PIN_LED LED_BUILTIN

#define SEALEVELPRESSURE_HPA (1013.25)

Scheduler scheduler;

void blinkCb();
void blinkOffCb();

Task taskBlink(500, TASK_FOREVER, &blinkCb, &scheduler, false, NULL,
&blinkOffCb);

uint8_t blinkOn;

Servo *servoArray;
int servoArraySize;

Adafruit_BMP3XX *bmp390Array;
int bmp390ArraySize;
TwoWire I2C = TwoWire(0);

BNO08x *bno08xArray;
int bno08xArraySize;
SPIClass vspi = SPIClass(VSPI);
TwoWire I2C1 = TwoWire(1);

int *pinArray;
int pinArraySize;
```

```
BLECharacteristic *pCharBlink;
BLECharacteristic *pServo;
BLECharacteristic *pBMP390;
BLECharacteristic *pBNO08X;
BLECharacteristic *pPin;

void (*resetFunc)(void) = 0;  // create a standard reset function

void setBlink(bool on, bool notify = false) {
  if (blinkOn == on) return;

  blinkOn = on;
  if (blinkOn) {
    Serial.println("Blink ON");
    taskBlink.restartDelayed(0);
  } else {
    Serial.println("Blink OFF");
    taskBlink.disable();
  }

  pCharBlink->setValue(&blinkOn, 1);
  if (notify) {
    pCharBlink->notify();
  }
}

void blinkCb() {
  digitalWrite(PIN_LED, taskBlink.getRunCounter() & 1);
```

```
}

void blinkOffCb() {
  digitalWrite(PIN_LED, 0);
}


class MyServerCallbacks : public BLEServerCallbacks {
  void onConnect(BLEServer *pServer) {
    Serial.println("Connected");
  };

  void onDisconnect(BLEServer *pServer) {
    Serial.println("Disconnected");
    resetFunc();
  }
};

class BlinkCallbacks : public BLECharacteristicCallbacks {
  void onWrite(BLECharacteristic *pCharacteristic) {
    String value = pCharacteristic->getValue();

    if (value.length() == 1) {
      uint8_t v = value[0];
      Serial.print("Got blink value: ");
      Serial.println(v);
      setBlink(v ? true : false);
      if (v) {
        pCharacteristic->setValue("On");
```

```
      } else {
        pCharacteristic->setValue("Off");
      }
      pCharacteristic->notify();
    } else {
      Serial.println("Invalid data received");
    }
  }
};

class ServoCallbacks : public BLECharacteristicCallbacks {
  void onWrite(BLECharacteristic *pCharacteristic) {
    String value = pCharacteristic->getValue();
    if (value.substring(0, 1) == "0") {
      servoArraySize = value.substring(1, value.length()).toInt();
      servoArray = new Servo[servoArraySize];
      Serial.println(servoArraySize);
      Serial.println("Initializing servos");
    }
    if (value.substring(0, 1) == "1") {
      for (int i = 0; i < servoArraySize; i++) {
        servoArray[i].attach(value.substring(2 * i + 1, 2 * i + 3).toInt());
        Serial.println(value.substring(2 * i + 1, 2 * i + 3).toInt());
      }
      Serial.println("Attaching servos");
    }
    if (value.substring(0, 1) == "2") {
      servoArray[value.substring(1, 3).toInt()].write(value.substring(3,
value.length()).toInt());
```

```
      Serial.println(value.substring(1, 3).toInt());
      Serial.println(value.substring(3, value.length()).toInt());
      Serial.println("Writing servos");
    }
    Serial.println(value);
  }
};

class BMP390Callbacks : public BLECharacteristicCallbacks {
  void onWrite(BLECharacteristic *pCharacteristic) {
    String value = pCharacteristic->getValue();
    if (value.substring(0, 1) == "0") {
      bmp390ArraySize = value.substring(1, value.length()).toInt();
      bmp390Array = new Adafruit_BMP3XX[bmp390ArraySize];
      Serial.println("Allocating bmp390s");
    }
    if (value.substring(0, 1) == "1") {
      //beginI2C: int address, twowire *wire
      if (value.length() == 7) {
        int SDA = value.substring(3, 5).toInt();
        int SCL = value.substring(5, 7).toInt();
        I2C.begin(SDA, SCL, 100000);
        bmp390Array[value.substring(1, 3).toInt()].begin_I2C(0x77, &I2C);
        bmp390Array[value.substring(1,
3).toInt()].setTemperatureOversampling(BMP3_OVERSAMPLING_8X);
        bmp390Array[value.substring(1,
3).toInt()].setPressureOversampling(BMP3_OVERSAMPLING_4X);
        bmp390Array[value.substring(1,
3).toInt()].setIIRFilterCoeff(BMP3_IIR_FILTER_COEFF_3);
```

```
      bmp390Array[value.substring(1,
3).toInt()].setOutputDataRate(BMP3_ODR_50_HZ);
    }
    //beginSPI: CS, SCK, MISO, MOSI
    else {
      bmp390Array[value.substring(1, 3).toInt()].begin_SPI(value.substring(3, 5).toInt(),
value.substring(5, 7).toInt(), value.substring(7, 9).toInt(), value.substring(9, 11).toInt());
      bmp390Array[value.substring(1,
3).toInt()].setTemperatureOversampling(BMP3_OVERSAMPLING_8X);
      bmp390Array[value.substring(1,
3).toInt()].setPressureOversampling(BMP3_OVERSAMPLING_4X);
      bmp390Array[value.substring(1,
3).toInt()].setIIRFilterCoeff(BMP3_IIR_FILTER_COEFF_3);
      bmp390Array[value.substring(1,
3).toInt()].setOutputDataRate(BMP3_ODR_50_HZ);
    }
    Serial.println("Attaching bmp390s");
  }
  if (value.substring(0, 1) == "2") {
    String bmp390Number = value.substring(1, 3);
    bmp390Array[bmp390Number.toInt()].performReading();
    Serial.println("Writing bmp390s");
    pCharacteristic->setValue(bmp390Number + "3" +
String(bmp390Array[value.substring(1, 3).toInt()].temperature));
    pCharacteristic->notify();
    pCharacteristic->setValue(bmp390Number + "4" +
String(bmp390Array[value.substring(1, 3).toInt()].pressure));
    pCharacteristic->notify();
    pCharacteristic->setValue(bmp390Number + "5" +
String(bmp390Array[value.substring(1,
3).toInt()].readAltitude(SEALEVELPRESSURE_HPA)));
    pCharacteristic->notify();
```

```
    }
    // Serial.println(value);
  }
};

class BNO08XCallbacks : public BLECharacteristicCallbacks {
  void onWrite(BLECharacteristic *pCharacteristic) {
    String value = pCharacteristic->getValue();
    if (value.substring(0, 1) == "0") {
      bno08xArraySize = value.substring(1, value.length()).toInt();
      bno08xArray = new BNO08x[bno08xArraySize];
      Serial.println("Allocating bno08x");
    }
    if (value.substring(0, 1) == "1") {
      if (value.length() == 7) {
        int SDA = value.substring(3, 5).toInt();
        int SCL = value.substring(5, 7).toInt();
        I2C1.begin(SDA, SCL, 100000);
        bno08xArray[value.substring(1, 3).toInt()].begin(0x4B, I2C1, -1, -1);
        bno08xArray[value.substring(1, 3).toInt()].enableRotationVector();
        bno08xArray[value.substring(1, 3).toInt()].enableGyro();
        delay(100);
        bno08xArray[value.substring(1, 3).toInt()].enableAccelerometer();
        Serial.println("began i2c");
      }
      //spi.begin: SCK, MISO, MOSI, CS
      else {
        vspi.begin(value.substring(3, 5).toInt(), value.substring(5, 7).toInt(),
value.substring(7, 9).toInt(), value.substring(9, 11).toInt());
```

```
      bno08xArray[value.substring(1, 3).toInt()].beginSPI(value.substring(9, 11).toInt(),
value.substring(11, 13).toInt(), value.substring(13, 15).toInt(), 1000000, vspi);
      delay(50);
      setReports(value.substring(1, 3).toInt());
      Serial.println("Attaching bno08xs");
    }
  }
  if (value.substring(0, 1) == "2") {
    Serial.println("Received");
    String bno08xNumber = value.substring(1, 3);
    int bno08xNumberInt = bno08xNumber.toInt();
    if (bno08xArray[bno08xNumberInt].wasReset()) {
      Serial.println("sensor was reset");
      setReports(bno08xNumberInt);
    }
    if (bno08xArray[bno08xNumberInt].getSensorEvent() == true) {}
    float quatI = bno08xArray[bno08xNumberInt].getQuatI();
    float quatJ = bno08xArray[bno08xNumberInt].getQuatJ();
    float quatK = bno08xArray[bno08xNumberInt].getQuatK();
    float quatReal = bno08xArray[bno08xNumberInt].getQuatReal();
    float quatAccuracy = bno08xArray[bno08xNumberInt].getQuatRadianAccuracy();

    float xGyro = bno08xArray[bno08xNumberInt].getGyroX();
    float yGyro = bno08xArray[bno08xNumberInt].getGyroY();
    float zGyro = bno08xArray[bno08xNumberInt].getGyroZ();

    float xAccelerometer = bno08xArray[bno08xNumberInt].getAccelX();
    float yAccelerometer = bno08xArray[bno08xNumberInt].getAccelY();
    float zAccelerometer = bno08xArray[bno08xNumberInt].getAccelZ();
```

```
      pCharacteristic->setValue(bno08xNumber + "3" + String(quatI, 2) + "," +
String(quatJ, 2) + "," + String(quatK, 2) + "," + String(quatReal, 2) + "," +
String(quatAccuracy, 2));

      pCharacteristic->notify();

      Serial.println(bno08xNumber + "3" + String(quatI, 2) + "," + String(quatJ, 2) + "," +
String(quatK, 2) + "," + String(quatReal, 2) + "," + String(quatAccuracy, 2));


      pCharacteristic->setValue(bno08xNumber + "4" + String(xGyro, 2) + "," +
String(yGyro, 2) + "," + String(zGyro, 2));

      pCharacteristic->notify();

      Serial.println(bno08xNumber + "4" + String(xGyro, 2) + "," + String(yGyro, 2) + "," +
String(zGyro, 2));


      pCharacteristic->setValue(bno08xNumber + "5" + String(xAccelerometer, 2) + "," +
String(yAccelerometer, 2) + "," + String(zAccelerometer, 2));

      pCharacteristic->notify();

      Serial.println(bno08xNumber + "4" + String(xAccelerometer, 2) + "," +
String(yAccelerometer, 2) + "," + String(zAccelerometer, 2));


      Serial.println("notifying bno08x");

    }

    Serial.println(value);

  }

};


class PinCallbacks : public BLECharacteristicCallbacks {

  void onWrite(BLECharacteristic *pCharacteristic) {

    String value = pCharacteristic->getValue();

    if (value.substring(0, 1) == "0") {

      pinArraySize = value.substring(1, value.length()).toInt();
```

```
      pinArray = new int[pinArraySize];
      Serial.println("Allocating bno08x");
    }
    if (value.substring(0, 1) == "1") {
      pinMode(value.substring(1, 3).toInt(), OUTPUT);
    }
    if (value.substring(0, 1) == "2") {
      if(value.substring(3,4) == "0") {
        digitalWrite(value.substring(1, 3).toInt(), LOW);
      }
      else {
        digitalWrite(value.substring(1, 3).toInt(), HIGH);
      }
    }
    if (value.substring(0, 1) == "3") {
      analogWrite(value.substring(1, 3).toInt(), value.substring(3, 6).toInt());
    }
    Serial.println(value);
  }
};


void setReports(int bno08xNumberInt) {
  if (bno08xArray[bno08xNumberInt].enableRotationVector() == true) {
    Serial.println(F("Rotation vector enabled"));
    Serial.println(F("Output in form i, j, k, real, accuracy"));
  } else {
    Serial.println("Could not enable rotation vector");
```

```
  }
  delay(100);

  if (bno08xArray[bno08xNumberInt].enableGyro() == true) {
    Serial.println(F("Gyro enabled"));
    Serial.println(F("Output in form x, y, z, in radians per second"));
  } else {
    Serial.println("Could not enable gyro");
  }
  delay(100);

  if (bno08xArray[bno08xNumberInt].enableAccelerometer() == true) {
    Serial.println(F("Accelerometer enabled"));
  } else {
    Serial.println("Could not enable accelerometer");
  }
  delay(100);
}

void setup() {
  Serial.begin(115200);
  Serial.println("Starting...");

  pinMode(PIN_LED, OUTPUT);

  String devName = DEVICE_NAME;
  String chipId = String((uint32_t)(ESP.getEfuseMac() >> 24), HEX);
  devName += '_';
```

```
  devName += chipId;

  BLEDevice::init(devName.c_str());
  BLEServer *pServer = BLEDevice::createServer();
  pServer->setCallbacks(new MyServerCallbacks());

  BLEService *pService = pServer->createService(SERVICE_UUID);

  pCharBlink = pService->createCharacteristic(BLINK_UUID,
BLECharacteristic::PROPERTY_READ | BLECharacteristic::PROPERTY_NOTIFY |
BLECharacteristic::PROPERTY_WRITE);
  pCharBlink->setCallbacks(new BlinkCallbacks());

  pServo = pService->createCharacteristic(SERVO_UUID,
BLECharacteristic::PROPERTY_READ | BLECharacteristic::PROPERTY_NOTIFY |
BLECharacteristic::PROPERTY_WRITE);
  pServo->setCallbacks(new ServoCallbacks());

  pBMP390 = pService->createCharacteristic(BMP390_UUID,
BLECharacteristic::PROPERTY_READ | BLECharacteristic::PROPERTY_NOTIFY |
BLECharacteristic::PROPERTY_WRITE);
  pBMP390->setCallbacks(new BMP390Callbacks());

  pBNO08X = pService->createCharacteristic(BNO08X_UUID,
BLECharacteristic::PROPERTY_READ | BLECharacteristic::PROPERTY_NOTIFY |
BLECharacteristic::PROPERTY_WRITE);
  pBNO08X->setCallbacks(new BNO08XCallbacks());

  pService->start();

  BLEAdvertising *pAdvertising = pServer->getAdvertising();
```

```
  BLEAdvertisementData adv;
  adv.setName(devName.c_str());
  pAdvertising->setAdvertisementData(adv);

  BLEAdvertisementData adv2;
  adv2.setCompleteServices(BLEUUID(SERVICE_UUID));
  pAdvertising->setScanResponseData(adv2);

  pAdvertising->start();

  Serial.println("Ready");
  Serial.print("Device name: ");
  Serial.println(devName);
}

void loop() {
  scheduler.execute();
}
```