

```
*
TheRecieverCode.ino

This program uses the ArduinoBLE library to set-up an Arduino Nano 33 BLE
as a peripheral device and specifies a service and a characteristic. Depending
of the value of the specified characteristic, one of two actuators is controlled,
indexing up and down.

This program also uses the NanoBLEFlashPrefs library to save the position of the
actuators late in the pages of flash space of the Arudino Nano to make small
space of non-volatile memory so that the actuators do not initialize at a position
that they are not at when the device is powered on and off.

The circuit:
- Arduino Nano 33 BLE.
- 2 Actuonix linear actuators

THIS IS FOR REVEICER ARDUINO (ACTUATOR)
this receives a byte of data named gesture (should be only transmitted when the 'gesture' is differnt from the last one)

*/

#include <ArduinoBLE.h>
#include <Servo.h>
#include <NanoBLEFlashPrefs.h>
Servo myServo;
Servo myServo2;
#define PIN_SERVO (5) // change to 5 for it to work
#define PIN_SERVO2 (7) // change to 6 to make work

const char* deviceServiceUuid = "19b10000-e8f2-537e-4f6c-d104768a1214";
const char* deviceServiceCharacteristicUuid = "19b10001-e8f2-537e-4f6c-d104768a1214";

int d = 10;
int i = 50;
int i2 = 50;
int VRy = 0;
int gesture = 0;

//struct for flash storage method (stores i, or position of actuator 1)
typedef struct positionStruct {
  uint8_t actuatorpos; // 0-255 avalibile, 1 byte
} positionprefs;
positionprefs prefs;
NanoBLEFlashPrefs myFlashPrefs;

BLEService gestureService(deviceServiceUuid);
BLEByteCharacteristic gestureCharacteristic(deviceServiceCharacteristicUuid, BLERead | BLEWrite);

void setup() {
  myServo.attach(PIN_SERVO);
  myServo2.attach(PIN_SERVO2);

  Serial.begin(9600);
  delay(5000);

  Serial.println("Read record...");
  int rc = myFlashPrefs.readPrefs(&prefs, sizeof(prefs));
  if (rc == FDS_SUCCESS)
  {
    Serial.println("Preferences found: ");
    Serial.println(prefs.actuatorpos);
    i = prefs.actuatorpos;
  }
  else
  {
    Serial.print("No preferences found. Return code: ");
    Serial.print(rc);
    Serial.print(", ");
    Serial.println(myFlashPrefs.errorString(rc));
  }

  if (!BLE.begin()) {
    Serial.println("- Starting Bluetooth® Low Energy module failed!");
    while (1);
  }

  BLE.setLocalName("Arduino Nano 33 BLE (Peripheral)");
  BLE.setAdvertisedService(gestureService);
  gestureService.addCharacteristic(gestureCharacteristic);
  BLE.addService(gestureService);
  gestureCharacteristic.writeValue(-1);
  BLE.advertise();

  Serial.println("Nano 33 BLE (Peripheral Device)");
  Serial.println(" ");

  myFlashPrefs.garbageCollection();
}

void loop() {
  BLEDevice central = BLE.central();
  Serial.println("- Discovering central device...");
  delay(500);

  if (central) {
    Serial.println("* Connected to central device!");
    Serial.print("* Device MAC address: ");
    Serial.println(central.address());
    Serial.println(" ");

    while (central.connected()) {
      if (gestureCharacteristic.written()) {
        gesture = gestureCharacteristic.value();
        writeGesture(gesture);
      }
    }

    Serial.println("* Disconnected to central device!");
  }
}

void SetStrokePerc(float strokePercentage)
{
  if ( strokePercentage >= 1.0 && strokePercentage <= 99.0 ) // clamps stroke percentage 1-99 so no actuator strain
  {
    int usec = 1000 + strokePercentage * ( 2000 - 1000 ) / 100.0 ;
    myServo.writeMicroseconds( usec );
  }
}

void SetStrokePerc2(float strokePercentage)
{
  if ( strokePercentage >= 1.0 && strokePercentage <= 99.0 ) // clamps stroke percentage 1-99 so no actuator strain
  {
    int usec = 1000 + strokePercentage * ( 2000 - 1000 ) / 100.0 ;
    myServo2.writeMicroseconds( usec );
  }
}

void SetStrokeMM(int strokeReq, int strokeMax)
{
  SetStrokePerc( ((float)strokeReq) / strokeMax );
}

void writeGesture(int gesture) {
  Serial.println(" - Characteristic <gesture_type> has changed!");
  if (i > 100) { //limits i vals so we dont have i=300 etc.
    i = 100;
  }
  if (i < 0) {
    i = 0;
  }
  if (gesture == 0) // if joystick is in the downshift position(ie, p = 0)
  {
    i -= d; // subtracts d from stroke percentage of actuator
    SetStrokePerc(i); // sets to the new stroke percentage
    delay(100); // delay so you get a shift and not just continuous motion
  }
  if (gesture == 2) // if the joystick is below its mapped value of 55, then the actuator position is increased by d
  {
    i += d; // adds d from stroke percentage of actuator
    SetStrokePerc(i); // sets to the new stroke percentage
    delay(100); // delay so you get a shift and not just continuous motion
  }
  if (gesture == 3)
  {
    //SetStrokePerc2(i2 - d); // needs to be writing to the other actuator - move one way
    delay(2000); // delay so you get a shift and not just continuous motion
    //SetStrokePerc2(i2);
    Serial.println("Dropper post!"); // comes back to where it was different pin than setstroke perc
  }

  SetStrokePerc(i);
  Serial.print("i=");
  Serial.print(i);

  prefs.actuatorpos = i;
  Serial.println("Write ActuatorPos...");
  myFlashPrefs.deletePrefs();
  myFlashPrefs.writePrefs(&prefs, sizeof(prefs));
}
```