```
  // Biggest value for int is 32767
// commented out blueString lines - with them the code does not run and they do nothing
//use this with Swift program AdafruitTest in Swift3 folder///
//okay with both BLE and SD cards -
#include <SD.h>
#include <SPI.h>

// SJK  added - note pin conflicts with original

#include "Adafruit_BLE_UART.h"
#include <Adafruit_BLE_Firmata.h>

// Connect CLK/MISO/MOSI to hardware SPI
// e.g. On UNO & compatible: CLK = 13, MISO = 12, MOSI = 11

#define ADAFRUITBLE_REQ 10
#define ADAFRUITBLE_RDY 2     // This should be an interrupt pin, on Uno thats #2 or #3
#define ADAFRUITBLE_RST 9
#define RESET_PIN 8
// so we have digital 3-7 and analog 0-6


Adafruit_BLE_UART BLEserial = Adafruit_BLE_UART(ADAFRUITBLE_REQ,
ADAFRUITBLE_RDY, ADAFRUITBLE_RST);


// make one instance for the user to use
Adafruit_BLE_FirmataClass BLE_Firmata(BLEserial);
/
*=========================================================================
========
 * GLOBAL VARIABLES

*=========================================================================
======*/

/////int BlueIntens[] = {0,2500,10,2500}; // Intensity out of 3000
//int CycleTimes[] = {120,120,120,120}; // Times depends on the duration set in blueon below.
default - minutes
/////int CycleTimes[] = {0,0,0,0};
//int Cycles = sizeof(CycleTimes)/sizeof(CycleTimes[0]);
int BlueIntens[10];
int CycleTimes[10];
long Cycles = 0;
aci_evt_opcode_t status;
aci_evt_opcode_t laststatus = ACI_EVT_DISCONNECTED;
String __concat = "";
bool done = false;
bool readyToGo = false;
String exptName = "";
// end SJK addition

File dataFile; //SJK omit for testing
int lightPin0 = 8;//define a pin for Photo resistor
```

```
int lightPin1 = 9;
int lightPin2 = 10;
int lightPin3 = 11;
int lightPin4 = 12;//define a pin for Photo resistor
int lightPin5 = 13;
int lightPin6 = 14;
int lightPin7 = 15;
static const uint8_t analog_pins[] =
{lightPin0,lightPin1,lightPin2,lightPin3,lightPin4,lightPin5,lightPin6,lightPin7}; // SJK added
int sensorValue = 0; // SJK added
// SJK must change pin int Rled0 = 9;
int Rled0 = 30;
int Rled1 = 32;
int Rled2 = 34;
int Rled3 = 36;
int Rled4 = 38;
int Rled5 = 40;
int Rled6 = 42;
int Rled7 = 44;
int Bled0 = 31;
int Bled1 = 33;
int Bled2 = 35;
int Bled3 = 37;
int Bled4 = 39;
int Bled5 = 41;
int Bled6 = 43;
int Bled7 = 45;
int Tr0=0; //percent transmission
int Tr1=0;
int Tr2=0;
int Tr3=0;
int Tr4=0;
int Tr5=0;
int Tr6=0;
int Tr7=0;
int off=0;
int j = 0;
int passes = 0;
const int chipSelect = 8;
String Filename = ""; //whatever you want the file containing all the data from the run to be
called NO CAPITALS!!!!!!
#include <EEPROM.h>

void setup() {
  // put your setup code here, to run once:
  digitalWrite(RESET_PIN, HIGH);
  delay(200);
  // initialize the digital pin as an output.
  //pinMode(led, OUTPUT);
  pinMode(RESET_PIN, OUTPUT);
 Serial.begin(9600);

 // SJK added section
  while(!Serial); // Leonardo/Micro should wait for serial init
```

```
  Serial.println(F("Adafruit Bluefruit Low Energy nRF8001 combo"));
  BLEserial.begin(); // SJK added
 // SJK end added section

  pinMode(Rled0, OUTPUT);
  pinMode(Rled1, OUTPUT);
  pinMode(Rled2, OUTPUT);
  pinMode(Rled3, OUTPUT);
  pinMode(Rled4, OUTPUT);
  pinMode(Rled5, OUTPUT);
  pinMode(Rled6, OUTPUT);
  pinMode(Rled7, OUTPUT);

  pinMode(Bled0, OUTPUT);
  pinMode(Bled1, OUTPUT);
  pinMode(Bled2, OUTPUT);
  pinMode(Bled3, OUTPUT);
  pinMode(Bled4, OUTPUT);
  pinMode(Bled5, OUTPUT);
  pinMode(Bled6, OUTPUT);
  pinMode(Bled7, OUTPUT);


// SJK omit so I can test - no SD card
    if (!SD.begin(chipSelect)) {
    Serial.println(F("Card failed, or not present"));
    // don't do anything more:
    return;
    } //SJK end omit
}

void loop() {
  // SJK added section
  // Tell the nRF8001 to do whatever it should be working on.
  BLEserial.pollACI();

  // Ask what is our current status
  status = BLEserial.getState();
  // If the status changed....
  if (status != laststatus) {
    // print it out!
   // Serial.println(F("**** STATUS CHANGED ****"));
    if (status == ACI_EVT_DEVICE_STARTED) {
        Serial.println(F("* Advertising started"));
    }
    if (status == ACI_EVT_CONNECTED) {
        Serial.println(F("* Connected!"));
    }
    if (status == ACI_EVT_DISCONNECTED) {
        Serial.println(F("* Disconnected or advertising timed out"));
    }
    // OK set the last status change to this one
    laststatus = status;
  }
```

```
 parseInputString();

 // SJK end added section
 if ((status == ACI_EVT_CONNECTED) && (readyToGo == true)) { // SJK added this line
 // put your main code here, to run repeatedly:
 Serial.println(F("Collecting data."));
 Serial.print("We have ");Serial.print(Cycles); Serial.println(" Cycles");
 for (int i=0; i < Cycles; i++){ // number of different light intensities
  Serial.print(F("Cycle index"));Serial.println(i);
  for (int z=0; z < CycleTimes[i]; z++){ // number of minutes at the chosen intensity
    Serial.print(F("CycleTimes index "));Serial.println(z);
    blueon(BlueIntens[i], 20000); // blueon(intesity, duration) duration 20000 = 60 seconds
    // SJK shortened to 2000 for testing - normal is 20000
    measure();
   }
 }
 //while(1);
 readyToGo = false;

 } //SJK added to end if clause

}




void blueon(int intensity, int duration) {
      j = 0;
      while (j <=duration)  {
      if (intensity>0)
      {
      digitalWrite(Bled0, HIGH);
      digitalWrite(Bled1, HIGH);
      digitalWrite(Bled2, HIGH);
      digitalWrite(Bled3, HIGH);
      digitalWrite(Bled4, HIGH);
      digitalWrite(Bled5, HIGH);
      digitalWrite(Bled6, HIGH);
      digitalWrite(Bled7, HIGH);
      delayMicroseconds(intensity); // Approximately .05% duty cycle @ 1KHz
      digitalWrite(Bled0, LOW);
      digitalWrite(Bled1, LOW);
      digitalWrite(Bled2, LOW);
      digitalWrite(Bled3, LOW);
      digitalWrite(Bled4, LOW);
      digitalWrite(Bled5, LOW);
      digitalWrite(Bled6, LOW);
      digitalWrite(Bled7, LOW);
      delayMicroseconds(3000-intensity);
      }
            else
      {
        digitalWrite(Bled0, LOW);
        digitalWrite(Bled1, LOW);
```

```
        digitalWrite(Bled2, LOW);
        digitalWrite(Bled3, LOW);
        digitalWrite(Bled4, LOW);
        digitalWrite(Bled5, LOW);
        digitalWrite(Bled6, LOW);
        digitalWrite(Bled7, LOW);
        delayMicroseconds(3000);
       }
       j=j+1;
   }
}

void measure() {
  if (digitalRead(Bled0) == 1) {
     digitalWrite(Bled0, LOW);
     digitalWrite(Bled1, LOW);
     digitalWrite(Bled2, LOW);
     digitalWrite(Bled3, LOW);
     digitalWrite(Bled4, LOW);
     digitalWrite(Bled5, LOW);
     digitalWrite(Bled6, LOW);
     digitalWrite(Bled7, LOW);
     }
     delay(90);
     digitalWrite(Rled0, HIGH);
     digitalWrite(Rled1, HIGH);
     digitalWrite(Rled2, HIGH);
     digitalWrite(Rled3, HIGH);
     digitalWrite(Rled4, HIGH);
     digitalWrite(Rled5, HIGH);
     digitalWrite(Rled6, HIGH);
     digitalWrite(Rled7, HIGH);
     delayMicroseconds(100);
      // SJK added section
     for (int i = 0; i < 8; i++) {
            sensorValue = analogRead(analog_pins[i]);
            Serial.print(F("Analog")); Serial.print(analog_pins[i]); Serial.print(F(" = "));
Serial.println(sensorValue);
            BLE_Firmata.sendAnalog(analog_pins[i],sensorValue);
            BLEserial.pollACI(); //THIS IS MAGIC LINE - now it works
         }
         Serial.println(F("**********************************"));
     // SJK end added section

     Tr0=analogRead(lightPin0);
     Tr1=analogRead(lightPin1);
     Tr2=analogRead(lightPin2);
     Tr3=analogRead(lightPin3);
     Tr4=analogRead(lightPin4);
     Tr5=analogRead(lightPin5);
     Tr6=analogRead(lightPin6);
     Tr7=analogRead(lightPin7);

     digitalWrite(Rled0, LOW);
```

```
      digitalWrite(Rled1, LOW);
      digitalWrite(Rled2, LOW);
      digitalWrite(Rled3, LOW);
      digitalWrite(Rled4, LOW);
      digitalWrite(Rled5, LOW);
      digitalWrite(Rled6, LOW);
      digitalWrite(Rled7, LOW);

     // SJK omit since I have no SD card

      dataFile = SD.open(Filename, FILE_WRITE);
      dataFile.print(float(millis())/float(60000));
      dataFile.print(',');
      dataFile.print(Tr0);
      dataFile.print(',');
      dataFile.print(Tr1);
      dataFile.print(',');
      dataFile.print(Tr2);
      dataFile.print(',');
      dataFile.print(Tr3);
      dataFile.print(',');
      dataFile.print(Tr4);
      dataFile.print(',');
      dataFile.print(Tr5);
      dataFile.print(',');
      dataFile.print(Tr6);
      dataFile.print(',');
      dataFile.println(Tr7);
      dataFile.close();
      Serial.print(F("done"));// end omit



}

void parseInputString(void) {
  if (status == ACI_EVT_CONNECTED) {
    // Lets see if there's any data for us!
    if (BLEserial.available()) {
      Serial.print("* "); Serial.print(BLEserial.available()); Serial.println(F(" bytes available from
BLE"));
    }
    // OK while we still have something to read, get a character and print it out

    while (BLEserial.available()) {
     char c = BLEserial.read();
        if ((c != 'X') && (done != true)) {
          __concat = __concat + c;
          Serial.println(__concat);
        } else {
          done = true;
        }
    }
```

```
    if (done == true) {
      int INPUT_SIZE = 100;
      // Get next command from Serial (add 1 for final %)
      char input[INPUT_SIZE + 1];
      __concat.getBytes((byte*)input,INPUT_SIZE); // turn __concat into byte array input *** cast
needed because one library assumes signed and another assumes unsigned
      int size = sizeof(input);
      // Add the final % to end the C string
      input[size] = '%';

      // Read each command
      char* command = strtok(input, "&");
      while (command != 0)
      {
         Serial.println(command);
         // This will work - will send data like "B,12,34,56,78&L,12,34,56,7&C,4X "
         // first letter will trigger if clause, and then numbers will feed into array
         // need & as separator and X to mark end
         String code = getValue(command, ',', 0);
         Serial.print("Code is ");Serial.println(code);
         if (code == "C") {
           Cycles = getValue(command, ',', 1).toInt();
           Serial.print("We have ");Serial.print(Cycles); Serial.println(" Cycles");
         } else if (code == "T") {
           for (int i = 0; i < Cycles; i++) {
             CycleTimes[i] = getValue(command, ',', i + 1).toInt();
             Serial.print("We have time ");Serial.print(CycleTimes[i]); Serial.println(" time");
           }
         } else if (code == "B") {
           for (int i = 0; i < Cycles; i++) {
            // String blueString = getValue(command, ',', i + 1);
            // Serial.print("Blue string is  ");Serial.println(blueString);
             BlueIntens[i] = getValue(command, ',', i + 1).toInt();
             Serial.print("We have blue value ");Serial.print(BlueIntens[i]); Serial.println(" intensity");
           }
         } else if (code == "*") {

            exptName = getValue(command, ',', 1);
            Filename = exptName + ".txt";
            Serial.print("The name is ");Serial.println(Filename);

         }

         // Find the next command in input string
         command = strtok(0, "&");
      }
      done = false; //reset so only loops once
      readyToGo = true;
     // Serial.print("Ready to go with ");Serial.print(Cycles); Serial.println(" Cycles");
   }

  }
}
String getValue(String data, char separator, int index)
```

```
{
    int found = 0;
    int strIndex[] = { 0, -1 };
    int maxIndex = data.length() - 1;

    for (int i = 0; i <= maxIndex && found <= index; i++) {
        if (data.charAt(i) == separator || i == maxIndex) {
            found++;
            strIndex[0] = strIndex[1] + 1;
            strIndex[1] = (i == maxIndex) ? i+1 : i;
        }
    }
    return found > index ? data.substring(strIndex[0], strIndex[1]) : "";
}
```