```
#include <SPI.h>
#include <SD.h>
#define PR_PIN A0// ფოტორეზისტორი
#define PB1_PIN A1// ღილაკი (ინფრაწითელი...)
#define PB2_PIN A2// ღილაკი (თეთრი...)
#define LED1_PIN 6// წითელი შუქდიოდი
#define LED2_PIN 3// თეთრი შუქდიოდი
#define PB3_PIN 8// ღილაკი (ინდიკატორები...)
#define LED1_INDICATOR_PIN 5// ინდიკატორი (მწვანე შუქდიოდი) (ინფრაწითელი...)
#define LED2_INDICATOR_PIN 9// ინდიკატორი (მწვანე შუქდიოდი) (თეთრი...)
#define SW_PIN 7// ჩამრთველი (რეჟიმის შეცვლა (ავტომატური და ხელით მართვადი)...)
#define SPK_PIN 4// ხმამაღლამოლაპარაკე...
#define MQ135_PIN A4// გაზის სენსორი (ჰაერის ხარისხის)...
#define bluetoothVcc 2// ბლუთუზის მოდულის კვება...
#define SW2_PIN A3
File values;
boolean mSD;
int PR_VALUE;
boolean PB1_STATE;
boolean PB2_STATE;
boolean MODE;
int a1;
int a2;
boolean b;
boolean LAST_MODE;
boolean LED1_ENABLE;
boolean LED2_ENABLE;
boolean PB3_STATE;
int a3;
boolean ENABLE;
byte LED1_VALUE = 255;
```

```
byte LED2_VALUE = 255;
boolean c;
unsigned long t;
boolean BLINK_LEDS_ENABLE;
int LED1_INDICATOR_VALUE;
int LED2_INDICATOR_VALUE;
boolean PB1_STATE_2;
boolean PB2_STATE_2;
boolean CHANGE_VALUE_ENABLE;
int a4;
boolean LED1_INDICATOR_STATE;
boolean LED2_INDICATOR_STATE;
boolean LED1;
boolean LED2;
boolean d;
boolean lastPB1state;
boolean lastPB2state;
boolean lastPB3state;
boolean lastSWstate;
boolean PB1_VALUE;
boolean PB2_VALUE;
boolean PB3_VALUE;
boolean SW_VALUE;
unsigned long t2;
boolean PB1value;
boolean PB2value;
boolean PB3value;
boolean SWvalue;
boolean e;
boolean LAST_PB3_STATE;
boolean f;
```

```
int a5;
boolean g;
int a6;
boolean a7;
boolean a8;
int a9;
boolean functionEnable;
unsigned long t3;
unsigned long t4;
unsigned long t5;
unsigned long t6;
byte a1_2;
byte a2_2;
byte a3_2;
byte morseUnit = 150;
unsigned long t7;
byte b2;
boolean SOS_ENABLE;
boolean STROBE_ENABLE;
byte count;
byte count2;
byte last_count2;
boolean LED2_STATE;
byte b3;
unsigned long t8;
boolean SPK_STATE;
boolean SETUP;
int SPK_VALUE;
unsigned long t9;
boolean b4;
byte siren;
```

```
int NTCT_VALUE;
double NTCT_RESISTANCE;
double V_OUT_A1;
double a = -0.03155737752;
double R0 = 10954.65;
double TEMPERATURE0 = 290.25;
double DELTA_RESISTANCE;
double DELTA_TEMPERATURE;
double TEMPERATURE;
double R_INTERNAL_PULLUP_VALUE = 26276.6;
unsigned long t10;
byte second;
byte minute;
byte hour;
byte day;
byte month;
int year;
unsigned long T;
unsigned long t11;
boolean INDICATOR_ENABLE;
boolean SIREN_AND_SOS_SETUP;
boolean b5;
unsigned long t12;
boolean b6 = 1;
boolean LED1_INDICATOR_STATE_2;
boolean LED2_INDICATOR_STATE_2;
boolean b7;
boolean LED2_STATE_2;
boolean TONE_STATE;
unsigned long t13;
double value;
```

```
int potValue;
int lastPotValue;
unsigned long T2;
boolean A;
byte countA;
boolean B;
byte countB;
unsigned long t14;
int MQ135_VALUE;
double lowValue;
double mediumValue;
double highValue;
boolean INDICATORS_SIGNAL_F_E;
boolean BLINK_LEDS_SETUP;
boolean I_S_E;
boolean C;
unsigned long t15;
String bluetoothCommand;
double T_MQ = 6.45;
double V_OUT_A2;
double MQ135_RESISTANCE;
byte buttonsState;
boolean LED1state;
boolean LED2state;
unsigned long t16;
boolean DISABLE;
boolean I_E = 1;;
boolean CLOCK_ENABLE;
unsigned long t17;
String bluetoothID_IC = "IC4LED1SPK1PR1GS1SD1T";
String bluetoothID_APP = "APP4LED1SPK1PR1GS1SD1T";
```

```
String readID;
boolean D;
boolean b8;
String command;
unsigned long t18;
unsigned long t19;
unsigned long t20;
boolean LS1;
boolean LS2;
String test;
boolean E;
String BMODE = "PB";
boolean BLED1;
boolean BLED2;
boolean BSOS;
boolean BSTR;
boolean SETUP_DISABLE;
boolean SETUP_DISABLE1;
boolean SETUP_DISABLE2;
boolean BT;
String option;
byte valueOfLED;
boolean d1;
boolean d2;
boolean d3;
boolean d4;
boolean d5;
unsigned long t21;
unsigned long t22;
byte yearIndex2;
byte monthIndex1;
```

```
byte monthIndex2;

byte dayIndex1;

byte dayIndex2;

byte hourIndex1;

byte hourIndex2;

byte minuteIndex1;

byte minuteIndex2;

byte secondIndex1;

unsigned long t23;

unsigned long t24;

boolean bluetoothEnable;

int DV = 900;// (dark value) ფოტორეზისტორის პინზე არსებული ძაბვის შესაბამისი რიცხვი "სიბნელე"-ში

int FDV = 1005;// (full dark value) ფოტორეზისტორის პინზე არსებული ძაბვის შესაბამისი რიცხვი "სრული სიბნელე"-ში

String fileName;

boolean bluetoothEnable_STATE;

boolean values_closed = 1;

byte N;

boolean Value;

boolean VALUE[5];

boolean lastState[5];

boolean lastSW2state;

boolean SW2value;

boolean SW2_VALUE;

long Size;

boolean powerSave;

boolean ps;

unsigned long t25;


void enable(int n, boolean state)  {// ავტომატურ რეჟიმში, ინფრაწითელი და თეთრი შუქდიოდის მართვა (გამორთვა, ჩართვა...)...
```

```
 if (n == 1)  {
  if (state == 0)  {
   LED1_ENABLE = 0;
   analogWrite(LED1_INDICATOR_PIN, 0);
  }
  else if (state == 1)  {
   LED1_ENABLE = 1;
   if (ENABLE)
    analogWrite(LED1_INDICATOR_PIN, 128);
   else
    analogWrite(LED1_INDICATOR_PIN, 0);
  }
 }
 else if (n == 2)  {
  if (state == 0)  {
   LED2_ENABLE = 0;
   analogWrite(LED2_INDICATOR_PIN, 0);
  }
  else if (state == 1) {
   LED2_ENABLE = 1;
   if (ENABLE)
    analogWrite(LED2_INDICATOR_PIN, 128);
   else
    analogWrite(LED2_INDICATOR_PIN, 0);
  }
 }
}

boolean debounce(String component)  {// რხევამაქრის ფუნქცია
 if (component == "PB1") {
  PB1value = digitalRead(PB1_PIN);
```

```
  if (PB1value != lastPB1state)
    t2 = millis();
  if ((millis() - t2) > 50)
    PB1_VALUE = !PB1value;
  lastPB1state = PB1value;
  return PB1_VALUE;
}
else if (component == "PB2") {
  PB2value = digitalRead(PB2_PIN);
  if (PB2value != lastPB2state)

    t2 = millis();
  if ((millis() - t2) > 50)
    PB2_VALUE = !PB2value;
  lastPB2state = PB2value;
  return PB2_VALUE;
}
else if (component == "PB3") {
  PB3value = digitalRead(PB3_PIN);
  if (PB3value != lastPB3state)
    t2 = millis();
  if ((millis() - t2) > 50)
    PB3_VALUE = !PB3value;
  lastPB3state = PB3value;
  return PB3_VALUE;
}
else if (component == "SW") {
  SWvalue = digitalRead(SW_PIN);
  if (SWvalue != lastSWstate)
    t2 = millis();
  if ((millis() - t2) > 50)
```

```
    SW_VALUE = SWvalue;
   lastSWstate = SWvalue;
   return SW_VALUE;
  }
  else if (component == "SW2") {
   SW2value = digitalRead(SW2_PIN);
   if (SW2value != lastSW2state)
    t2 = millis();
   if ((millis() - t2) > 50)
    SW2_VALUE = SW2value;
   lastSW2state = SW2value;
   return SW2_VALUE;
  }
}
```

void PB3andLEDs(String LEDn) {// შუქდიოდების ნათების სიმძლავრის რეგულირებისას ინფრაწითელი ან თეთრი შუქდიოდის ანთება/ჩაქრობა...

```
  if (PB3_STATE == 1 & a6 == 0)  {
   a6 = 1;
  }
  else if (PB3_STATE == 0 & a6 == 1) {
   a6 = 2;
  }
  else if (PB3_STATE == 1 & a6 == 2) {
   analogWrite(LED1_PIN, 0);
   analogWrite(LED2_PIN, 0);
   a6 = 3;
  }
  else if (PB3_STATE == 0 & a6 == 3) {
   a6 = 0;
  }
```

```
  if (a6 == 2) {
    if (LEDn == "LED1")
      analogWrite(LED1_PIN, LED1_VALUE);
    else if (LEDn == "LED2")
      analogWrite(LED2_PIN, LED2_VALUE);
  }
}

void INDICATOR_PR(boolean INDICATOR_PR_ENABLE, boolean state1, boolean state2) {// ავტომატურ რეჟიმში, ინდიკატორების მართვა...
  if (INDICATOR_PR_ENABLE) {
    if (state1)
      analogWrite(LED1_INDICATOR_PIN, 128);
    else if (LED1_ENABLE)
      analogWrite(LED1_INDICATOR_PIN, 1);
    else
      analogWrite(LED1_INDICATOR_PIN, 0);
    if (state2)
      analogWrite(LED2_INDICATOR_PIN, 128);
    else if (LED2_ENABLE)
      analogWrite(LED2_INDICATOR_PIN, 1);
    else
      analogWrite(LED2_INDICATOR_PIN, 0);
  }
  else  {
    analogWrite(LED1_INDICATOR_PIN, 0);
    analogWrite(LED2_INDICATOR_PIN, 0);
  }
}

void INDICATOR_PB() {// ხელით მართვად რეჟიმში, ინდიკატორების მართვა...
```

```
 if (a1 == 2 | a1 == 3)
  analogWrite(LED1_INDICATOR_PIN, 128);
 else
  analogWrite(LED1_INDICATOR_PIN, 1);
 if (a2 == 2 | a2 == 3)
  analogWrite(LED2_INDICATOR_PIN, 128);
 else
  analogWrite(LED2_INDICATOR_PIN, 1);
}

void leds(boolean state) {
 digitalWrite(LED2_PIN, state);
 if (INDICATOR_ENABLE)
  analogWrite(LED2_INDICATOR_PIN, map(state, 0, 1, 0, 128));
}

void morse(byte n)  {
 if (millis() - t7 >= n * morseUnit)  {
  count2++;
  t7 = millis();
 }
}
/*
 void MORSE_UNIT_SPACE() {
 if (millis() - t7 >= morseUnit)  {
  digitalWrite(LED2_PIN, 1);
  if (INDICATOR_ENABLE)
   analogWrite(LED2_INDICATOR_PIN, 128);
  count2++;
  t7 = millis();
 }
```

```
}

void MORSE_LETTER_SPACE() {
if (millis() - t7 >= 3 * morseUnit)  {
  digitalWrite(LED2_PIN, 1);
  if (INDICATOR_ENABLE)
    analogWrite(LED2_INDICATOR_PIN, 128);
  count2++;
  t7 = millis();
}
}

void MORSE_WORD_SPACE()  {
if (millis() - t7 >= 7 * morseUnit)  {
  digitalWrite(LED2_PIN, 1);
  if (INDICATOR_ENABLE)
    analogWrite(LED2_INDICATOR_PIN, 128);
  count2++;
  t7 = millis();
}
}

void MORSE_1_UNIT() {
if (millis() - t7 >= morseUnit)  {
  digitalWrite(LED2_PIN, 0);
  analogWrite(LED2_INDICATOR_PIN, 0);
  count2++;
  t7 = millis();
}
}
```

```
 void MORSE_3_UNIT() {
 if (millis() - t7 >= 3 * morseUnit)  {
  digitalWrite(LED2_PIN, 0);
  analogWrite(LED2_INDICATOR_PIN, 0);
  count2++;
  t7 = millis();
 }
 }
*/
void morseSOS(boolean SPK, boolean INDICATOR_ENABLE_2) {
 INDICATOR_ENABLE = INDICATOR_ENABLE_2;
 if (count == 0)  {
  digitalWrite(LED2_PIN, 0);
  if(INDICATOR_ENABLE)
    digitalWrite(LED2_INDICATOR_PIN, 0);
  if (SPK)
    noTone(SPK_PIN);
  count = 1;
 }
 else if (count == 1)  {
  leds(0);
  morse(7);
  if (SPK)
    noTone(SPK_PIN);
 }
 else if (count == 2)  {
  leds(1);
  morse(1);
  if (SPK)
    tone(SPK_PIN, SPK_VALUE);
 }
```

```
else if (count == 3)  {
  leds(0);
  morse(1);
  if (SPK)
    noTone(SPK_PIN);
}
else if (count == 4)  {
  leds(1);
  morse(1);
  if (SPK)
    tone(SPK_PIN, SPK_VALUE);
}
else if (count == 5)  {
  leds(0);
  morse(1);
  if (SPK)
    noTone(SPK_PIN);
}
else if (count == 6)  {
  leds(1);
  morse(1);
  if (SPK)
    tone(SPK_PIN, SPK_VALUE);
}
else if (count == 7)  {
  leds(0);
  morse(3);
  if (SPK)
    noTone(SPK_PIN);
}
else if (count == 8)  {
```

```
  leds(1);
  morse(3);
  if (SPK)
    tone(SPK_PIN, SPK_VALUE);
}
else if (count == 9)  {
  leds(0);
  morse(1);
  if (SPK)
    noTone(SPK_PIN);
}
else if (count == 10) {
  leds(1);
  morse(3);
  if (SPK)
    tone(SPK_PIN, SPK_VALUE);
}
else if (count == 11) {
  leds(0);
  morse(1);
  if (SPK)
    noTone(SPK_PIN);
}
else if (count == 12) {
  leds(1);
  morse(3);
  if (SPK)
    tone(SPK_PIN, SPK_VALUE);
}
else if (count == 13) {
  leds(0);
```

```
  morse(3);
  if (SPK)
    noTone(SPK_PIN);
}
else if (count == 14) {
  leds(1);
  morse(1);
  if (SPK)
    tone(SPK_PIN, SPK_VALUE);
}
else if (count == 15) {
  leds(0);
  morse(1);
  if (SPK)
    noTone(SPK_PIN);
}
else if (count == 16) {
  leds(1);
  morse(1);
  if (SPK)
    tone(SPK_PIN, SPK_VALUE);
}
else if (count == 17) {
  leds(0);
  morse(1);
  if (SPK)
    noTone(SPK_PIN);
}
else if (count == 18) {
  leds(1);
  morse(1);
```

```
    if (SPK)
      tone(SPK_PIN, SPK_VALUE);
  }
  if (last_count2 != count2)  {
    if (count == 18)
      count = 0;
    count++;
    last_count2 = count2;
  }
}

void SET_DATE_AND_TIME(String option, int value)  {
  if (option == "SECOND")
    second = value;
  else if (option == "MINUTE")
    minute = value;
  else if (option == "HOUR")
    hour = value;
  else if (option == "DAY")
    day = value;
  else if (option == "MONTH")
    month = value;
  else if (option == "YEAR")
    year = value;
}

/*double AIR_QUALITY()  {
  MQ135_VALUE = analogRead(MQ135_PIN);
  V_OUT_A2 = (MQ135_VALUE / 1023.0) * 5.0;
  MQ135_RESISTANCE = 20000 * (1.0 / ((5.0 / V_OUT_A2) - 1.0));
  value = (MQ135_RESISTANCE / 200000.0) * 100;
```

```
  return value;
  }*/

double AIR_QUALITY()  {
  MQ135_VALUE = analogRead(MQ135_PIN);
  value = (MQ135_VALUE / 1023.0) * 100;
  return value;
}

void RECEIVE_DATE_AND_TIME(String dateAndTime)  {
  yearIndex2 = dateAndTime.indexOf("-", 0);
  monthIndex1 = yearIndex2 + 1;
  monthIndex2 = dateAndTime.indexOf("-", monthIndex1 + 1);
  dayIndex1 = monthIndex2 + 1;
  dayIndex2 = dateAndTime.indexOf(" ", dayIndex1 + 1);
  hourIndex1 = dayIndex2 + 1;
  hourIndex2 = dateAndTime.indexOf(":", hourIndex1 + 1);
  minuteIndex1 = hourIndex2 + 1;
  minuteIndex2 = dateAndTime.indexOf(":", minuteIndex1 + 1);
  secondIndex1 = minuteIndex2 + 1;
  SET_DATE_AND_TIME("YEAR", (dateAndTime.substring(0, yearIndex2)).toInt());
  SET_DATE_AND_TIME("MONTH", (dateAndTime.substring(monthIndex1, monthIndex2)).toInt());
  SET_DATE_AND_TIME("DAY", (dateAndTime.substring(dayIndex1, dayIndex2)).toInt());
  SET_DATE_AND_TIME("HOUR", (dateAndTime.substring(hourIndex1, hourIndex2)).toInt());
  SET_DATE_AND_TIME("MINUTE", (dateAndTime.substring(minuteIndex1, minuteIndex2)).toInt());
  SET_DATE_AND_TIME("SECOND", (dateAndTime.substring(secondIndex1)).toInt());
}

void delayFunction(byte delayTime) {
  for (t23 = millis(); millis() - t23 < delayTime; )
    DATE_AND_TIME();
```

```
}

long printDirectory(File dir, boolean mode) {
  while (true) {
    if (CLOCK_ENABLE)
      DATE_AND_TIME();
    File entry =  dir.openNextFile();
    if (! entry) {
      // no more files
      dir.rewindDirectory();
      entry.rewindDirectory();
      if (!mode)
        return Size;
      else if (mode)
        break;
    }
    //for (uint8_t i = 0; i < numTabs; i++) {
    //Serial.print('\t');
    //}
    if (!mode)
      Size += entry.size();
    else if (mode)
      Serial.print(entry.name());
    //if (entry.isDirectory()) {
    //Serial.println("/");
    //printDirectory(entry, numTabs + 1);
    //} else {
    // files have sizes, directories do not
    //Serial.print("\t\t");
    //Serial.println(entry.size(), DEC);
    //}
```

```
    entry.close();
  }
}

void setFileName()  {
  fileName = String(year);
  if (month < 10)
    fileName += "0";
  fileName += String(month);
  if (day < 10)
    fileName += "0";
  fileName += String(day);
}

void bluetooth()  {
  if (!E) {
    if (Serial.available()) {
      readID = Serial.readString();
      E = 1;
    }
  }
  if (readID == bluetoothID_APP)  {
    if (!D)  {
      Serial.print(bluetoothID_IC);
      D = 1;
    }
    if (b8 == 0)  {
      analogWrite(LED1_INDICATOR_PIN, 128);
      analogWrite(LED2_INDICATOR_PIN, 128);
    }
    if (Serial.available())  {
```

```
bluetoothCommand = Serial.readString();
option = bluetoothCommand.substring(0, 3);
if (!CLOCK_ENABLE && bluetoothCommand.length() >= 14)  {
  RECEIVE_DATE_AND_TIME(bluetoothCommand);
  CLOCK_ENABLE = 1;
}
if (bluetoothCommand == "VAL" && !powerSave) {
  Serial.print(String(AIR_QUALITY() * 1000000));

}
else if (bluetoothCommand == "RES") {
  LED1_VALUE = 255;
  LED2_VALUE = 255;
  DV = 900;
  FDV = 1005;
}
else if (bluetoothCommand == "LUX") {
  Serial.print(1023 - PR_VALUE);
}
else if (bluetoothCommand == "DIS")  {
  bluetooth_2(0);
  delayFunction(100);
  bluetooth_2(1);
}
else if (bluetoothCommand == "S3") {
  Serial.print(BLED1);
  Serial.print("|");
  Serial.print(BLED2);
  Serial.print("|");
  Serial.print(LED1_VALUE);
  Serial.print("|");
```

```
  Serial.print(LED2_VALUE);
  Serial.print("|");
  Serial.print(BMODE);
  Serial.print("|");
  Serial.print(BSOS);
  Serial.print("|");
  Serial.print(BSTR);
  if (LED1_VALUE < 100)
    Serial.print("|");
  if (LED1_VALUE < 10)
    Serial.print("|");
  if (LED2_VALUE < 100)
    Serial.print("|");
  if (LED2_VALUE < 10)
    Serial.print("|");
 }
 else if (bluetoothCommand == "S4" && !powerSave) {
  Size = 0;
  values.close();
  values = SD.open("/");
  Serial.print(printDirectory(values, 0));
  values.rewindDirectory();
  values.close();
  setFileName();
  values = SD.open(fileName + ".txt", FILE_WRITE);
 }
 else if (bluetoothCommand == "get" && !powerSave)  {
  values.close();
  values = SD.open("/");
  printDirectory(values, 1);
  values.rewindDirectory();
```

```
  values.close();
  setFileName();
  values = SD.open(fileName + ".txt", FILE_WRITE);
 }
 else if (option == "SEL" && !powerSave) {
  values.close();
  values = SD.open(bluetoothCommand.substring(3, bluetoothCommand.length()));
  while (values.available()) {
    if (CLOCK_ENABLE)
      DATE_AND_TIME();
    Serial.write(values.read());
  }
  values.close();
  setFileName();
  values = SD.open(fileName + ".txt", FILE_WRITE);
 }
 else if (bluetoothCommand == "MPB")  {
  BMODE = "PB";
 }
 else if (bluetoothCommand == "MPR") {
  BMODE = "PR";
 }
 else if (bluetoothCommand == "1L1")  {
  BLED1 = 1;
  if (BMODE == "PB")
    BLED2 = 0;
 }
 else if (bluetoothCommand == "1L0") {
  BLED1 = 0;
 }
 else if (bluetoothCommand == "2L1")  {
```

```
    BLED2 = 1;
    if (BMODE == "PB")
      BLED1 = 0;
  }
  else if (bluetoothCommand == "2L0") {
    BLED2 = 0;
  }
  else if (bluetoothCommand == "SO1")  {
    BSOS = 1;
    BSTR = 0;
  }
  else if (bluetoothCommand == "SO0") {
    BSOS = 0;
    SETUP_DISABLE1 = 0;
    SETUP_DISABLE2 = 0;
  }
  else if (bluetoothCommand == "ST1")  {
    BSTR = 1;
    BSOS = 0;
  }
  else if (bluetoothCommand == "ST0") {
    BSTR = 0;
    SETUP_DISABLE1 = 0;
    SETUP_DISABLE2 = 0;
  }
  setLightValues();
}
if (BSOS) {
  SOS(1);
  d1 = 0;
}
```

```
   else if (!BSTR && !d1)  {
     SOS(0);
     d1 = 1;
   }
   if (BSTR) {
     STROBE(1);
     d2 = 0;
   }
   else if (!BSOS && !d2) {
     STROBE(0);
     d2 = 1;
   }
   if (!BSOS && !BSTR && !b8)
     LEDS_AND_BLUETOOTH();
  }
  else  {
   if (millis() - t16 >= 1000)  {
     LED1state = !LED1state;
     LED2state = !LED2state;
     analogWrite(LED1_INDICATOR_PIN, map(LED1state, 0, 1, 0, 128));
     analogWrite(LED2_INDICATOR_PIN, map(LED2state, 0, 1, 0, 128));
     t16 = millis();
   }
  }
}


void setLightValues() {
  if (option == "LI1")  {
   if ((bluetoothCommand.substring(3)).toInt() >= 0 && (bluetoothCommand.substring(3)).toInt() <=
1023 && (bluetoothCommand.substring(3)).toInt() > 1023 - FDV &&
(bluetoothCommand.substring(3)).toDouble() == (bluetoothCommand.substring(3)).toInt())  {
     DV = 1023 - (bluetoothCommand.substring(3)).toInt();
```

```
    Serial.print("DV1");
   }
   else
    Serial.print("DV0");
  }
  if (option == "LI0")  {
   if ((bluetoothCommand.substring(3)).toInt() >= 0 && (bluetoothCommand.substring(3)).toInt() <=
1023 && (bluetoothCommand.substring(3)).toInt() < 1023 - DV &&
(bluetoothCommand.substring(3)).toDouble() == (bluetoothCommand.substring(3)).toInt()) {
    FDV = 1023 - (bluetoothCommand.substring(3)).toInt();
    Serial.print("FDV1");
   }
   else
    Serial.print("FDV0");
  }
}

void LEDS_AND_BLUETOOTH() {
  if (option == "1LV")  {
   valueOfLED = byte((bluetoothCommand.substring(3)).toInt());
   LED1_VALUE = valueOfLED;
   option = "0";
  }
  else if (option == "2LV")  {
   valueOfLED = byte((bluetoothCommand.substring(3)).toInt());
   LED2_VALUE = valueOfLED;
   option = "0";
  }
  if (BMODE == "PB") {
   if (!d4)  {
    BLED1 = 0;
    BLED2 = 0;
```

```
    d3 = 0;
    d4 = 1;
   }
   if (BLED1 == 1)
    analogWrite(LED1_PIN, LED1_VALUE);
   else if (BLED1 == 0)
    digitalWrite(LED1_PIN, 0);
   if (BLED2 == 1)
    analogWrite(LED2_PIN, LED2_VALUE);
   else if (BLED2 == 0)
    digitalWrite(LED2_PIN, 0);
  }
  else if (BMODE == "PR")  {
   if (!d3)  {
    BLED1 = 1;
    BLED2 = 1;
    d4 = 0;
    d3 = 1;
   }
   if (DV < PR_VALUE & PR_VALUE <= FDV)  {
    if (BLED1)  {
     analogWrite(LED1_PIN, LED1_VALUE);
     digitalWrite(LED2_PIN, 0);
    }
    else if (BLED2) {
     digitalWrite(LED1_PIN, 0);
     analogWrite(LED2_PIN, LED2_VALUE);
    }
   }
   else  {
    digitalWrite(LED1_PIN, 0);
```

```
    digitalWrite(LED2_PIN, 0);
   }
   if (FDV < PR_VALUE & BLED2)  {
    digitalWrite(LED1_PIN, 0);
    analogWrite(LED2_PIN, LED2_VALUE);
   }
   else if (FDV < PR_VALUE & BLED1 & !BLED2) {
    analogWrite(LED1_PIN, LED1_VALUE);
    digitalWrite(LED2_PIN, 0);
   }
 }
 else  {
  digitalWrite(LED1_PIN, 0);
  digitalWrite(LED2_PIN, 0);
 }
}

void DATE_AND_TIME()  {
 if (second >= 0 && second <= 60 && minute >= 0 && minute <= 60 && hour >= 0 && hour <= 24 &&
day >= 1 && day <= 32 && month >= 1 && month <= 13) {
   if (millis() - t10 >= 1000)  {
    second++;
    t10 = millis();
   }
   if (second == 60)  {
    second = 0;
    minute++;
   }
   else if (minute == 60)  {
    minute = 0;
    hour++;
```

```
   }
   else if (hour == 24) {
    hour = 0;
    day++;
   }
   else if ((day == 32 && month == 1) || (day == 29 && month == 2 && year % 4 > 0) || (day == 30 && month == 2 && year % 4 == 0) || (day == 32 && month == 3) || (day == 31 && month == 4) || (day == 32 && month == 5) || (day == 30 && month == 6) || (day == 32 && month == 7) || (day == 32 && month == 8) || (day == 31 && month == 9) || (day == 32 & month == 10) || (day == 31 && month == 11) || (day == 32 && month == 12)) {
    day = 1;
    month++;
   }
   else if (month == 13) {
    month = 1;
    year++;
   }
 }
}


void microSD()  {
 if (!powerSave)  {
  if (mSD) {
   if (values)  {
    if (millis() - t18 >= 250)  {
     if (CLOCK_ENABLE)  {
      if (++N == 4)  {
       values.print(hour);
       values.print(":");
       values.print(minute);
       values.print(":");
       values.println(second);
```

```
            values.println();
            N = 0;
           }
         }
         values.println(AIR_QUALITY());
         values.println();
         values.close();
         values_closed = 1;
         t18 = millis();
       }
     }
     if (CLOCK_ENABLE) {
       DATE_AND_TIME();
       setFileName();
       if (values_closed)  {
         values = SD.open(fileName + ".txt", FILE_WRITE);
         values_closed = 0;
       }
     }
     else  {
       if (values_closed)  {
         values = SD.open("numbers.txt", FILE_WRITE);
         values_closed = 0;
       }
     }
    }
  }
}

void SOS(boolean enable)  {
  if (enable)  {
```

```
  if (!SETUP_DISABLE1)  {
   SOS_SETUP();
   SETUP_DISABLE1 = 1;
  }
  if (PR_VALUE > DV)  {
   if (!BT)
    analogWrite(LED1_INDICATOR_PIN, 128);
   digitalWrite(LED1_PIN, 1);
  }
  else  {
   if (!BT)
    analogWrite(LED1_INDICATOR_PIN, 0);
   digitalWrite(LED1_PIN, 0);
  }
  morseSOS(1, !BT);
 }
 else  {
  digitalWrite(LED1_PIN, 0);
  digitalWrite(LED2_PIN, 0);
  digitalWrite(LED1_INDICATOR_PIN, 0);
  digitalWrite(LED2_INDICATOR_PIN, 0);
  noTone(SPK_PIN);
  SPK_VALUE = 0;
  t7 = millis();
  count = 0;
  count2 = 0;
  last_count2 = 0;
 }
}

void STROBE(boolean enable) {
```

```
if (enable)  {
 if (!SETUP_DISABLE2) {
  STROBE_SETUP();
  SETUP_DISABLE2 = 1;
 }
 if (PR_VALUE > DV)  {
  if (!BT)
   analogWrite(LED1_INDICATOR_PIN, 128);
  digitalWrite(LED1_PIN, 1);
 }
 else  {
  if (!BT)
   analogWrite(LED1_INDICATOR_PIN, 0);
  digitalWrite(LED1_PIN, 0);
 }
 if (millis() - t8 >= 100)  {
  LED2_STATE = !LED2_STATE;
  digitalWrite(LED2_PIN, LED2_STATE);
  analogWrite(LED2_INDICATOR_PIN, map(LED2_STATE, 0, 1, 0, 128));
  t8 = millis();
 }
 if (millis() - t14 >= 1) {
  if (SPK_VALUE < 4000 && !b4)
   SPK_VALUE += 25;
  if (SPK_VALUE == 4000)
   b4 = 1;
  if (SPK_VALUE > 1000 && b4)
   SPK_VALUE -= 25;
  if (SPK_VALUE == 1000)
   b4 = 0;
  t14 = millis();
```

```
    }
    tone(SPK_PIN, SPK_VALUE);
  }
  else  {
    digitalWrite(LED1_PIN, 0);
    digitalWrite(LED2_PIN, 0);
    digitalWrite(LED1_INDICATOR_PIN, 0);
    digitalWrite(LED2_INDICATOR_PIN, 0);
    digitalWrite(SPK_PIN, 0);
    noTone(SPK_PIN);
    SPK_VALUE = 0;
    LED2_STATE = 0;
    t8 = millis();
  }
}

void SOS_SETUP()  {
  digitalWrite(LED1_PIN, 0);
  digitalWrite(LED2_PIN, 0);
  digitalWrite(LED1_INDICATOR_PIN, 0);
  digitalWrite(LED2_INDICATOR_PIN, 0);
  SPK_VALUE = 2500;
  t7 = millis();
  count = 0;
  count2 = 0;
  last_count2 = 0;
}

void STROBE_SETUP() {
  digitalWrite(LED1_PIN, 0);
  digitalWrite(LED2_PIN, 0);
```

```
  digitalWrite(LED1_INDICATOR_PIN, 0);
  digitalWrite(LED2_INDICATOR_PIN, 0);
  LED2_STATE = 0;
  SPK_VALUE = 1000;
  t8 = millis();
}

void bluetooth_2(boolean mode)  {
  if (mode)  {
    digitalWrite(bluetoothVcc, 1);
    Serial.begin(115200);
    d1 = 0;
    d2 = 0;
    d3 = 0;
    BT = 1;
    SOS_ENABLE = 1;
    STROBE_ENABLE = 1;
    DISABLE = 1;
  }
  else  {
    Serial.end();
    digitalWrite(bluetoothVcc, 0);
    digitalWrite(LED1_INDICATOR_PIN, 0);
    digitalWrite(LED2_INDICATOR_PIN, 0);
    d1 = 0;
    d2 = 0;
    d3 = 0;
    BT = 0;
    SOS_ENABLE = 0;
    STROBE_ENABLE = 0;
    LED1state = 0;
```

```
    LED2state = 0;
    t16 = millis();
    DISABLE = 0;
    D = 0;
    readID = "0";
    E = 0;
  }
}

void PB3_AND_BLUETOOTH()  {
  if (!PB3_STATE)
    t24 = millis();
  if (millis() - t24 > 1000 && PB3_STATE)
    bluetoothEnable_STATE = 1;
  else
    bluetoothEnable = 0;
  if (bluetoothEnable_STATE && !PB3_STATE) {
    bluetoothEnable = 1;
    bluetoothEnable_STATE = 0;
  }
}

void setup()  {
  pinMode(PR_PIN, INPUT_PULLUP);
  pinMode(PB1_PIN, INPUT_PULLUP);
  pinMode(PB2_PIN, INPUT_PULLUP);
  pinMode(LED1_PIN, OUTPUT);
  pinMode(LED2_PIN, OUTPUT);
  pinMode(SW_PIN, INPUT_PULLUP);
  pinMode(LED1_INDICATOR_PIN, OUTPUT);
  pinMode(LED2_INDICATOR_PIN, OUTPUT);
```

```
 pinMode(PB3_PIN, INPUT_PULLUP);
 pinMode(MQ135_PIN, INPUT);
 pinMode(bluetoothVcc, OUTPUT);
 pinMode(SW2_PIN, INPUT);
 Serial.setTimeout(100);
 if (digitalRead(SW2_PIN)) {
  tone(SPK_PIN, 1000);
  if (SD.begin(10))  {
   noTone(SPK_PIN);
   mSD = 1;
  }
  else  {
   noTone(SPK_PIN);
   while (!debounce("PB1") && !debounce("PB2") && !debounce("PB3"))
    if (millis() - t20 >= 250) {
     LS1 = !LS1;
     LS2 = !LS2;
     analogWrite(LED1_INDICATOR_PIN, map(LS1, 0, 1, 0, 128));
     analogWrite(LED2_INDICATOR_PIN, map(LS2, 0, 1, 0, 128));
     t20 = millis();
    }
  }
 }
 t18 = millis();
 t19 = millis();
 t24 = millis();
}

void (* resetFunc) (void) = 0;

void loop() {
```

```
PR_VALUE = analogRead(PR_PIN);
PB1_STATE = debounce("PB1");
PB2_STATE = debounce("PB2");
PB3_STATE = debounce("PB3");
MODE = debounce("SW");
powerSave = !debounce("SW2");
if (millis() - t25 >= 100)  {
 if (powerSave && !ps)  {
   ps = 1;
 }
 else if (!powerSave && ps) {
   ps = 0;
   resetFunc();
 }
 t25 = millis();
}
microSD();
if (buttonsState == 0)
 PB3_AND_BLUETOOTH();
if (bluetoothEnable && buttonsState == 0)  {
 bluetooth_2(1);
 bluetoothEnable = 0;
 buttonsState = 1;
}
else if (!PB1_STATE && !PB2_STATE && !PB3_STATE && buttonsState == 1)  {
 buttonsState = 2;
}
else if ((PB1_STATE || PB2_STATE || PB3_STATE) && buttonsState == 2)  {
 buttonsState = 3;
}
else if (!PB1_STATE && !PB2_STATE && !PB3_STATE && buttonsState == 3) {
```

```
  bluetooth_2(0);
  buttonsState = 0;
}
if (buttonsState == 2)
  bluetooth();
if (PB1_STATE & PB3_STATE & b2 == 0 & !STROBE_ENABLE && DISABLE == 0)  {
  SOS_ENABLE = 1;
  b2 = 1;
}
else if ((!PB1_STATE & !PB3_STATE) & b2 == 1 && DISABLE == 0) {
  SOS_SETUP();
  b2 = 2;
}
else if ((PB1_STATE | PB2_STATE | PB3_STATE) & b2 == 2) {
  SOS(0);
  b2 = 3;
}
else if (!PB1_STATE & !PB2_STATE & !PB3_STATE & b2 == 3) {
  SOS_ENABLE = 0;
  SETUP_DISABLE1 = 0;
  b2 = 0;
}
if (b2 == 2) {
  SOS(1);
}
if (PB2_STATE & PB3_STATE & b3 == 0 & !SOS_ENABLE && DISABLE == 0)  {
  STROBE_ENABLE = 1;
  b3 = 1;
}
else if (!PB2_STATE & !PB3_STATE & b3 == 1 && DISABLE == 0) {
  STROBE_SETUP();
```

```
  b3 = 2;
}
else if ((PB1_STATE | PB2_STATE | PB3_STATE) & b3 == 2) {
  STROBE(0);
  b3 = 3;
}
else if (!PB1_STATE & !PB2_STATE & !PB3_STATE & b3 == 3) {
  STROBE_ENABLE = 0;
  SETUP_DISABLE2 = 0;
  b3 = 0;
}
if (b3 == 2)  {
  STROBE(1);
}
if (DISABLE == 1)  {
  b2 = 0;
  b3 = 0;
}
if (!SOS_ENABLE & !STROBE_ENABLE)  {
  if (!SETUP)  {
    noTone(SPK_PIN);
    digitalWrite(SPK_PIN, 0);
    SETUP = 1;
  }
  if ((PB1_STATE == 1 & PB2_STATE == 1) & (a4 == 0))  {
    e = 1;
    digitalWrite(LED1_PIN, 0);
    digitalWrite(LED2_PIN, 0);
    digitalWrite(LED1_INDICATOR_PIN, 0);
    digitalWrite(LED2_INDICATOR_PIN, 0);
    a4 = 1;
```

```
}
else if ((PB1_STATE == 0 & PB2_STATE == 0) & (a4 == 1)) {
  CHANGE_VALUE_ENABLE = 1;
  a4 = 2;
}
else if (((PB1_STATE == 1 & PB2_STATE == 1) & (a4 == 2)) | (f & (!LED1 & !LED2))) {
  CHANGE_VALUE_ENABLE = 0;
  c = 0;
  d = 0;
  a1_2 = a1;
  a2_2 = a2;
  a3_2 = a3;
  a1 = 0;
  a2 = 0;
  a3 = 0;
  a5 = 0;
  a6 = 0;
  a7 = 0;
  a8 = 0;
  a9 = 0;
  functionEnable = 0;
  analogWrite(LED1_PIN, 0);
  analogWrite(LED2_PIN, 0);
  digitalWrite(LED1_INDICATOR_PIN, 0);
  digitalWrite(LED2_INDICATOR_PIN, 0);
  LED1 = 0;
  LED2 = 0;
  LAST_PB3_STATE = 0;
  f = 0;
  a4 = 3;
}
```

```
else if ((PB1_STATE == 0 & PB2_STATE == 0) & (a4 == 3)) {
 if (!MODE) {
  a1 = a1_2;
  a2 = a2_2;
  a3 = a3_2;
 }
 e = 0;
 a4 = 0;
}
if (CHANGE_VALUE_ENABLE)  {
 if (LED1_INDICATOR_STATE)
  LED1_INDICATOR_VALUE = 128;
 else
  LED1_INDICATOR_VALUE = 0;
 if (LED2_INDICATOR_STATE)
  LED2_INDICATOR_VALUE = 128;
 else
  LED2_INDICATOR_VALUE = 0;
 if (!c)  {
  BLINK_LEDS_ENABLE = 1;
  t = millis();
  c = 1;
 }
 if ((millis() - t >= 500) & BLINK_LEDS_ENABLE)  {
  LED1_INDICATOR_STATE = !LED1_INDICATOR_STATE;
  LED2_INDICATOR_STATE = !LED2_INDICATOR_STATE;
  analogWrite(LED1_INDICATOR_PIN, LED1_INDICATOR_VALUE);
  analogWrite(LED2_INDICATOR_PIN, LED2_INDICATOR_VALUE);
  t = millis();
 }
 if ((PB3_STATE | LAST_PB3_STATE) & (!LED1 & !LED2)) {
```

```
  LAST_PB3_STATE = 1;
  analogWrite(LED1_INDICATOR_PIN, map(LED1_VALUE, 1, 255, 1, 128));
  analogWrite(LED2_INDICATOR_PIN, map(LED2_VALUE, 1, 255, 1, 128));
}
if (PB3_STATE & g == 0)  {
  a5++;
  g = 1;
}
else if (!PB3_STATE & g == 1) {
  g = 0;
}
else if (!PB3_STATE & a5 == 2) {
  g = 0;
  f = 1;
  a5 = 0;
}
if (!LAST_PB3_STATE & (!LED1 & !LED2))  {
  if (PB1_STATE & !a7)
    a7 = 1;
  else if (!PB1_STATE & a7)  {
    LED1 = 1;
    LED2 = 0;
    a7 = 0;
  }
  if (PB2_STATE & !a8)
    a8 = 1;
  else if (!PB2_STATE & a8)  {
    LED2 = 1;
    LED1 = 0;
    a8 = 0;
  }
```

```
}
if (LED1)  {
  if (d == 0)  {
    BLINK_LEDS_ENABLE = 0;
    analogWrite(LED1_INDICATOR_PIN, map(LED1_VALUE, 1, 255, 1, 128));
    analogWrite(LED2_INDICATOR_PIN, 0);
    d = 1;
  }
  PB3andLEDs("LED1");
  if (PB1_STATE & LED1_VALUE > 1) {
    if (millis() - t3 >= 10)  {
      LED1_VALUE--;
      analogWrite(LED1_INDICATOR_PIN, map(LED1_VALUE, 1, 255, 1, 128));
      t3 = millis();
    }
  }
  else if (PB2_STATE & LED1_VALUE < 255) {
    if (millis() - t4 >= 10)  {
      LED1_VALUE++;
      analogWrite(LED1_INDICATOR_PIN, map(LED1_VALUE, 1, 255, 1, 128));
      t4 = millis();
    }
  }
}
else if (LED2)  {
  if (d == 0)  {
    BLINK_LEDS_ENABLE = 0;
    analogWrite(LED1_INDICATOR_PIN, 0);
    analogWrite(LED2_INDICATOR_PIN, map(LED2_VALUE, 1, 255, 1, 128));
    d = 1;
  }
```

```
    PB3andLEDs("LED2");
    if (PB1_STATE & LED2_VALUE > 1) {
     if (millis() - t5 >= 10)  {
       LED2_VALUE--;
       analogWrite(LED2_INDICATOR_PIN, map(LED2_VALUE, 1, 255, 1, 128));
       t5 = millis();
     }
    }
    else if (PB2_STATE & LED2_VALUE < 255) {
     if (millis() - t6 >= 10)  {
       LED2_VALUE++;
       analogWrite(LED2_INDICATOR_PIN, map(LED2_VALUE, 1, 255, 1, 128));
       t6 = millis();
     }
    }
  }
}
if (!e)  {
  if (MODE == 0)  {
    if ((LED2_ENABLE & ENABLE))
      PB1_STATE_2 = PB1_STATE;
    if ((LED1_ENABLE & ENABLE))
      PB2_STATE_2 = PB2_STATE;
    if (b == 0)  {
      a1 = 0;
      a2 = 0;
      a3 = 0;
      ENABLE = 0;
      enable(1, 1);
      enable(2, 1);
      digitalWrite(LED1_PIN, 0);
```

```
  digitalWrite(LED2_PIN, 0);
  b = 1;
 }
 if (!functionEnable) {
  if ((PB1_STATE | PB2_STATE) & a3 == 0)  {
   a3 = 1;
  }
  else if ((!PB1_STATE & !PB2_STATE) & a3 == 1) {
   ENABLE = 1;
   a3 = 2;
  }
  else if (PB3_STATE & a3 == 2) {
   a3 = 3;
  }
  else if (!PB3_STATE & a3 == 3) {
   ENABLE = 0;
   a3 = 0;
  }
  if (PB1_STATE_2 == 0 & a1 == 0)  {
   enable(1, 1);
  }
  else if (PB1_STATE_2 == 1 & a1 == 0)  {
   a1 = 1;
  }
  else if (PB1_STATE_2 == 0 & a1 == 1)  {
   enable(1, 0);
   a1 = 2;
  }
  else if (PB1_STATE_2 == 1 & a1 == 2) {
   a1 = 3;
  }
```

```
  else if (PB1_STATE_2 == 0 & a1 == 3) {
    enable(1, 1);
    a1 = 0;
  }
  if (PB2_STATE_2 == 0 & a2 == 0)  {
    enable(2, 1);
  }
  else if (PB2_STATE_2 == 1 & a2 == 0)  {
    a2 = 1;
  }
  else if (PB2_STATE_2 == 0 & a2 == 1)  {
    enable(2, 0);
    a2 = 2;
  }
  else if (PB2_STATE_2 == 1 & a2 == 2) {
    a2 = 3;
  }
  else if (PB2_STATE_2 == 0 & a2 == 3) {
    enable(2, 1);
    a2 = 0;
  }
}
if (!ENABLE) {
  if ((PB3_STATE == 1) & (a9 == 0))  {
    a9 = 1;
  }
  else if ((PB3_STATE == 0) & (a9 == 1)) {
    functionEnable = 1;
    a9 = 2;
  }
  else if ((PB3_STATE == 1) & (a9 == 2)) {
```

```
 a9 = 3;
}
else if ((PB3_STATE == 0) & (a9 == 3)) {
 functionEnable = 0;
 analogWrite(LED1_INDICATOR_PIN, 0);
 analogWrite(LED2_INDICATOR_PIN, 0);
 a9 = 0;
}
if (a9 == 2)
 functionEnable = 1;
else
 functionEnable = 0;
if (DV < PR_VALUE & PR_VALUE <= FDV)  {
 if (LED1_ENABLE)  {
   INDICATOR_PR(functionEnable, 1, 0);
   analogWrite(LED1_PIN, LED1_VALUE);
   digitalWrite(LED2_PIN, 0);
 }
 else if (LED2_ENABLE) {
   INDICATOR_PR(functionEnable, 0, 1);
   digitalWrite(LED1_PIN, 0);
   analogWrite(LED2_PIN, LED2_VALUE);
 }
}
else  {
 INDICATOR_PR(functionEnable, 0, 0);
 digitalWrite(LED1_PIN, 0);
 digitalWrite(LED2_PIN, 0);
}
if (FDV < PR_VALUE & LED2_ENABLE == 1)  {
 INDICATOR_PR(functionEnable, 0, 1);
```

```
      digitalWrite(LED1_PIN, 0);
      analogWrite(LED2_PIN, LED2_VALUE);
    }
    else if (FDV < PR_VALUE & LED1_ENABLE == 1 & LED2_ENABLE == 0) {
      INDICATOR_PR(functionEnable, 1, 0);
      analogWrite(LED1_PIN, LED1_VALUE);
      digitalWrite(LED2_PIN, 0);
    }
  }
  else  {
    digitalWrite(LED1_PIN, 0);
    digitalWrite(LED2_PIN, 0);
  }
}
else {
  if (b == 1)  {
    a1 = 0;
    a2 = 0;
    a3 = 0;
    a9 = 0;
    functionEnable = 0;
    enable(1, 0);
    enable(2, 0);
    digitalWrite(LED1_PIN, 0);
    digitalWrite(LED2_PIN, 0);
    b = 0;
  }
  if (PB3_STATE == 1 & a3 == 0)  {
    a3 = 1;
  }
  else if (PB3_STATE == 0 & a3 == 1)  {
```

```
  INDICATOR_PB();
  a3 = 2;
}
else if (PB3_STATE == 1 & a3 == 2) {
  a3 = 3;
}
else if (PB3_STATE == 0 & a3 == 3) {
  analogWrite(LED1_INDICATOR_PIN, 0);
  analogWrite(LED2_INDICATOR_PIN, 0);
  a3 = 0;
}
if (a3 == 2 && I_E) {
  INDICATOR_PB();
}
if (PB1_STATE == 1 & a1 == 0)  {
  a1 = 1;
}
else if (PB1_STATE == 0 & a1 == 1)  {
  analogWrite(LED1_PIN, LED1_VALUE);
  digitalWrite(LED2_PIN, 0);
  a1 = 2;
  a2 = 0;
}
else if (PB1_STATE == 1 & a1 == 2) {
  a1 = 3;
  a2 = 0;
}
else if (PB1_STATE == 0 & a1 == 3) {
  digitalWrite(LED1_PIN, 0);
  digitalWrite(LED2_PIN, 0);
  a1 = 0;
```

```
      a2 = 0;
    }
    if (PB2_STATE == 1 & a2 == 0)  {
      a2 = 1;
    }
    else if (PB2_STATE == 0 & a2 == 1)  {
      digitalWrite(LED1_PIN, 0);
      analogWrite(LED2_PIN, LED2_VALUE);
      a2 = 2;
      a1 = 0;
    }
    else if (PB2_STATE == 1 & a2 == 2) {
      a2 = 3;
      a1 = 0;
    }
    else if (PB2_STATE == 0 & a2 == 3) {
      digitalWrite(LED1_PIN, 0);
      digitalWrite(LED2_PIN, 0);
      a2 = 0;
      a1 = 0;
    }
   }
  }
}
else  {
  if (MODE)  {
    a1 = 0;
    a2 = 0;
  }
  a3 = 0;
  a4 = 0;
```

```
    a5 = 0;
    a6 = 0;
    a7 = 0;
    a8 = 0;
    a9 = 0;
    SETUP = 0;
    CHANGE_VALUE_ENABLE = 0;
    c = 0;
    d = 0;
    functionEnable = 0;
    LED1 = 0;
    LED2 = 0;
    LAST_PB3_STATE = 0;
    f = 0;
    e = 0;
  }
}
```