////////////////////////////////////////////////////////////////////////

// Arduino Interface with Mindwave

//

// This is example code provided by NeuroSky, Inc. and is provided

// license free.

////////////////////////////////////////////////////////////////////////

#define LED 13

#define BAUDRATE 115200

#define DEBUGOUTPUT 0

#define GREENLED1 3

#define GREENLED2 4

#define GREENLED3 5

#define YELLOWLED1 6

#define YELLOWLED2 7

#define YELLOWLED3 8

#define YELLOWLED4 9

#define REDLED1 10

#define REDLED2 11

#define REDLED3 12

// checksum variables

byte generatedChecksum = 0;

byte checksum = 0;

int payloadLength = 0;

byte payloadData[64] = {

0};

byte poorQuality = 0;

byte attention = 0;

byte meditation = 0;

// system variables

long lastReceivedPacket = 0;

boolean bigPacket = false;

//////////////////////////

// Microprocessor Setup //

//////////////////////////

void setup() {

pinMode(GREENLED1, OUTPUT);

pinMode(GREENLED2, OUTPUT);

pinMode(GREENLED3, OUTPUT);

pinMode(YELLOWLED1, OUTPUT);

pinMode(YELLOWLED2, OUTPUT);

pinMode(YELLOWLED3, OUTPUT);

pinMode(YELLOWLED4, OUTPUT);

pinMode(REDLED1, OUTPUT);

pinMode(REDLED2, OUTPUT);

pinMode(REDLED3, OUTPUT);

pinMode(LED, OUTPUT);

Serial.begin(BAUDRATE); // USB

delay(3000) ;

Serial.print(194,BYTE) ;

}

////////////////////////////////

// Read data from Serial UART //

////////////////////////////////

byte ReadOneByte() {

int ByteRead;

while(!Serial.available());

ByteRead = Serial.read();

#if DEBUGOUTPUT

Serial.print((char)ByteRead); // echo the same byte out the USB serial (for debug purposes)

#endif

return ByteRead;

}

/////////////

//MAIN LOOP//

/////////////

void loop() {

// Look for sync bytes

if(ReadOneByte() == 170) {

if(ReadOneByte() == 170) {

payloadLength = ReadOneByte();

if(payloadLength > 169) //Payload length can not be greater than 169

return;

generatedChecksum = 0;

for(int i = 0; i < payloadLength; i++) {

payloadData[i] = ReadOneByte(); //Read payload into memory

generatedChecksum += payloadData[i];

}

checksum = ReadOneByte(); //Read checksum byte from stream

generatedChecksum = 255 - generatedChecksum; //Take one's compliment of generated checksum

if(checksum == generatedChecksum) {

poorQuality = 200;

attention = 0;

meditation = 0;

for(int i = 0; i < payloadLength; i++) { // Parse the payload

switch (payloadData[i]) {

case 2:

i++;

poorQuality = payloadData[i];

bigPacket = true;

break;

case 4:

i++;

attention = payloadData[i];

break;

case 5:

i++;

meditation = payloadData[i];

break;

case 0x80:

i = i + 3;

break;

case 0x83:

i = i + 25;

break;

default:

break;

} // switch

} // for loop

#if !DEBUGOUTPUT

// \*\*\* Add your code here \*\*\*

if(bigPacket) {

if(poorQuality == 0)

digitalWrite(LED, HIGH);

else

digitalWrite(LED, LOW);

Serial.print("PoorQuality: ");

Serial.print(poorQuality, DEC);

Serial.print(" Attention: ");

Serial.print(attention, DEC);

if(attention > 40)

digitalWrite(GREENLED1, HIGH);

else

digitalWrite(GREENLED1, LOW);

Serial.print(" Time since last packet: ");

Serial.print(millis() - lastReceivedPacket, DEC);

lastReceivedPacket = millis();

Serial.print("\n");

switch(attention / 10) {

case 0:

digitalWrite(GREENLED1, HIGH);

digitalWrite(GREENLED2, LOW);

digitalWrite(GREENLED3, LOW);

digitalWrite(YELLOWLED1, LOW);

digitalWrite(YELLOWLED2, LOW);

digitalWrite(YELLOWLED3, LOW);

digitalWrite(YELLOWLED4, LOW);

digitalWrite(REDLED1, LOW);

digitalWrite(REDLED2, LOW);

digitalWrite(REDLED3, LOW);

break;

case 1:

digitalWrite(GREENLED1, HIGH);

digitalWrite(GREENLED2, HIGH);

digitalWrite(GREENLED3, LOW);

digitalWrite(YELLOWLED1, LOW);

digitalWrite(YELLOWLED2, LOW);

digitalWrite(YELLOWLED3, LOW);

digitalWrite(YELLOWLED4, LOW);

digitalWrite(REDLED1, LOW);

digitalWrite(REDLED2, LOW);

digitalWrite(REDLED3, LOW);

break;

case 2:

digitalWrite(GREENLED1, HIGH);

digitalWrite(GREENLED2, HIGH);

digitalWrite(GREENLED3, HIGH);

digitalWrite(YELLOWLED1, LOW);

digitalWrite(YELLOWLED2, LOW);

digitalWrite(YELLOWLED3, LOW);

digitalWrite(YELLOWLED4, LOW);

digitalWrite(REDLED1, LOW);

digitalWrite(REDLED2, LOW);

digitalWrite(REDLED3, LOW);

break;

case 3:

digitalWrite(GREENLED1, HIGH);

digitalWrite(GREENLED2, HIGH);

digitalWrite(GREENLED3, HIGH);

digitalWrite(YELLOWLED1, HIGH);

digitalWrite(YELLOWLED2, LOW);

digitalWrite(YELLOWLED3, LOW);

digitalWrite(YELLOWLED4, LOW);

digitalWrite(REDLED1, LOW);

digitalWrite(REDLED2, LOW);

digitalWrite(REDLED3, LOW);

break;

case 4:

digitalWrite(GREENLED1, HIGH);

digitalWrite(GREENLED2, HIGH);

digitalWrite(GREENLED3, HIGH);

digitalWrite(YELLOWLED1, HIGH);

digitalWrite(YELLOWLED2, HIGH);

digitalWrite(YELLOWLED3, LOW);

digitalWrite(YELLOWLED4, LOW);

digitalWrite(REDLED1, LOW);

digitalWrite(REDLED2, LOW);

digitalWrite(REDLED3, LOW);

break;

case 5:

digitalWrite(GREENLED1, HIGH);

digitalWrite(GREENLED2, HIGH);

digitalWrite(GREENLED3, HIGH);

digitalWrite(YELLOWLED1, HIGH);

digitalWrite(YELLOWLED2, HIGH);

digitalWrite(YELLOWLED3, HIGH);

digitalWrite(YELLOWLED4, LOW);

digitalWrite(REDLED1, LOW);

digitalWrite(REDLED2, LOW);

digitalWrite(REDLED3, LOW);

break;

case 6:

digitalWrite(GREENLED1, HIGH);

digitalWrite(GREENLED2, HIGH);

digitalWrite(GREENLED3, HIGH);

digitalWrite(YELLOWLED1, HIGH);

digitalWrite(YELLOWLED2, HIGH);

digitalWrite(YELLOWLED3, HIGH);

digitalWrite(YELLOWLED4, HIGH);

digitalWrite(REDLED1, LOW);

digitalWrite(REDLED2, LOW);

digitalWrite(REDLED3, LOW);

break;

case 7:

digitalWrite(GREENLED1, HIGH);

digitalWrite(GREENLED2, HIGH);

digitalWrite(GREENLED3, HIGH);

digitalWrite(YELLOWLED1, HIGH);

digitalWrite(YELLOWLED2, HIGH);

digitalWrite(YELLOWLED3, HIGH);

digitalWrite(YELLOWLED4, HIGH);

digitalWrite(REDLED1, HIGH);

digitalWrite(REDLED2, LOW);

digitalWrite(REDLED3, LOW);

break;

case 8:

digitalWrite(GREENLED1, HIGH);

digitalWrite(GREENLED2, HIGH);

digitalWrite(GREENLED3, HIGH);

digitalWrite(YELLOWLED1, HIGH);

digitalWrite(YELLOWLED2, HIGH);

digitalWrite(YELLOWLED3, HIGH);

digitalWrite(YELLOWLED4, HIGH);

digitalWrite(REDLED1, HIGH);

digitalWrite(REDLED2, HIGH);

digitalWrite(REDLED3, LOW);

break;

case 9:

digitalWrite(GREENLED1, HIGH);

digitalWrite(GREENLED2, HIGH);

digitalWrite(GREENLED3, HIGH);

digitalWrite(YELLOWLED1, HIGH);

digitalWrite(YELLOWLED2, HIGH);

digitalWrite(YELLOWLED3, HIGH);

digitalWrite(YELLOWLED4, HIGH);

digitalWrite(REDLED1, HIGH);

digitalWrite(REDLED2, HIGH);

digitalWrite(REDLED3, HIGH);

break;

case 10:

digitalWrite(GREENLED1, HIGH);

digitalWrite(GREENLED2, HIGH);

digitalWrite(GREENLED3, HIGH);

digitalWrite(YELLOWLED1, HIGH);

digitalWrite(YELLOWLED2, HIGH);

digitalWrite(YELLOWLED3, HIGH);

digitalWrite(YELLOWLED4, HIGH);

digitalWrite(REDLED1, HIGH);

digitalWrite(REDLED2, HIGH);

digitalWrite(REDLED3, HIGH);

break;

}

}

#endif

bigPacket = false;

}

else {

// Checksum Error

} // end if else for checksum

} // end if read 0xAA byte

} // end if read 0xAA byte

}