```
#include <SPI.h>
#include <Wire.h>
#include <ss_oled.h>
#include <EEPROM.h>

#define GROVE_SDA_PIN A2
#define GROVE_SCL_PIN A3
#define SDA_PIN A4
#define SCL_PIN A5
#define RESET_PIN -1
#define OLED_ADDR -1
#define FLIP180 0
#define INVERT 0
#define USE_HW_I2C 0
#define MY_OLED1 OLED_128x64
#define MY_OLED2 OLED_128x64
#define PAUSE 3
#define TIMEOUT 4

SSOLED ssoled[2];

//unsigned long milisegundos;
unsigned long clock, ultimoclock, millisMax;
unsigned long segundosJugador[2];
unsigned long BigClock = 0;
int bonus;
int moves[2];
int movimientos;    // Contadores de movimientos
int player = PAUSE;
bool botonPresionado[2], timeout=false, link=false, displayed=false;


struct persistente{
  unsigned long tiempo;
  unsigned long bonus;
  int beep;
};

persistente persist;

void setup() {
    int inutil;
    inutil = oledInit(&ssoled[0], MY_OLED1, OLED_ADDR, FLIP180, INVERT, 1,
SDA_PIN, SCL_PIN, RESET_PIN, 400000L);
    inutil = oledInit(&ssoled[1], MY_OLED2, OLED_ADDR, FLIP180, INVERT, 0,
GROVE_SDA_PIN, GROVE_SCL_PIN, RESET_PIN, 400000L);

    Serial.begin(9600);

   EEPROM.get(0,persist);

 //   bonus=persist.bonus;

  //persist.tiempo=600;
```

```
  //persist.bonus=1;

  EEPROM.get(0,persist);
  //Serial.println(variableToString(persist.bonus));
  //Serial.println(variableToString(persist.tiempo));

  for(int i = 0; i < 2; i++) {
       segundosJugador[i] = persist.tiempo;
        bonus = persist.bonus;

      moves[i] = 0;
      botonPresionado[i] = 0;
      oledFill(&ssoled[i], 0, 1);
  }
  ultimoclock = 0;
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(2, INPUT_PULLUP);
  pinMode(3, INPUT_PULLUP);
  pinMode(4, INPUT_PULLUP);
  pinMode(5, INPUT_PULLUP);
  pinMode(6, INPUT_PULLUP);
  pinMode(A0, INPUT);
  oledFill(&ssoled[0], 0, 1);
  oledFill(&ssoled[1], 0, 1);
  digitalWrite(13, LOW);
  if ((persist.beep)==1){
  digitalWrite(7, HIGH);
  delay(25);
  digitalWrite(7, LOW);
  }
  timeSetting(0);
}

void beep(unsigned long time){
  if ((persist.beep)==1){
    millisMax = millis()+time;
    digitalWrite(7, HIGH);
    Serial.println(millis()+"  "+millisMax);
  }
}

void timeSetting(int type) {
    char buf[15];
    if (type == 0) { // I" "nitial setting when starting the clock
        oledFill(&ssoled[0], 0, 1);
        oledFill(&ssoled[1], 0, 1);
        oledWriteString(&ssoled[0], 0, 0, 0, (char *)"Confirm time settings",
FONT_SMALL, 0, 1);
        oledWriteString(&ssoled[0], 0, 0, 1, (char *)"retrieved from memory",
```

```
FONT_SMALL, 0, 1);
        oledWriteString(&ssoled[0], 0, 0, 7, (char *)"PAUSE: confirm",
FONT_NORMAL, 0, 1);
        oledWriteString(&ssoled[0], 0, 0, 6, (char *)"+/-: modify", FONT_NORMAL,
0, 1);
        oledWriteString(&ssoled[1], 0, 30, 0, (char *)"Time/Bonus", FONT_SMALL,
0, 1);
        ToHMS(segundosJugador[0],buf);
        oledWriteString(&ssoled[1], 0, 0, 2, (char *)buf, FONT_STRETCHED, 0, 1);
        ToHMS(bonus,buf);
        oledWriteString(&ssoled[1], 0, 0, 5, (char *)buf, FONT_STRETCHED, 0, 1);

      while (1) {
        if (digitalRead(5)== LOW){
          break;
        }
        if (digitalRead(4)== LOW){
          timeSetting(1);
        }
        if (digitalRead(6)== LOW){
          timeSetting(1);
        }
        if (Serial.available()) {
          String command = Serial.readStringUntil('\n');
          if (command == "LINK LOGON") {
          // Send LOGON-CONFIRM back to the Python side
          Serial.println("LOGON CONFIRM");
          link = true;
          break;
          }
        }
      }
      oledFill(&ssoled[0], 0, 1);
      oledFill(&ssoled[1], 0, 1);
    } else if (type == 1) {
      player = PAUSE;
      int mod = 0;
        oledFill(&ssoled[0], 0, 1);
        oledFill(&ssoled[1], 0, 1);
        oledWriteString(&ssoled[0], 0, 0, 0, (char *)"Default time settings",
FONT_SMALL, 0, 1);
        oledWriteString(&ssoled[0], 0, 0, 7, (char *)"PAUSE: next/conf.",
FONT_NORMAL, 0, 1);
        oledWriteString(&ssoled[0], 0, 0, 6, (char *)"+/-: modify", FONT_NORMAL,
0, 1);
        oledWriteString(&ssoled[1], 0, 30, 0, (char *)"Time/Bonus", FONT_SMALL,
0, 1);
        ToHMS(persist.tiempo,buf);
        oledWriteString(&ssoled[1], 0, 0, 2, (char *)buf, FONT_STRETCHED, 0, 1);
        ToHMS(persist.bonus,buf);
        oledWriteString(&ssoled[1], 0, 0, 5, (char *)buf, FONT_STRETCHED, 0, 1);
      while (1){
        ToHMS(persist.tiempo,buf);
        oledWriteString(&ssoled[1], 0, 0, 2, (char *)buf, FONT_STRETCHED, 0, 1);
```

```
        ToHMS(persist.bonus,buf);
        oledWriteString(&ssoled[1], 0, 0, 5, (char *)buf, FONT_STRETCHED, 0, 1);
        if (digitalRead(4)== LOW){
          if (mod == 0){
            persist.tiempo+=600;
          } else if (mod == 1){
            persist.bonus++;
          }
        }
        if (digitalRead(5)== LOW){
          if (mod == 0){
            mod++;
          } else if (mod == 1){
            break;
          }
          else{
            Serial.println("ERROR: parametro incorrecto");
          }
        }
        if (digitalRead(6)== LOW){
          if (mod == 0){
            persist.tiempo-=600;
          } else if (mod == 1){
            persist.bonus--;
          }
        }
      }
        oledFill(&ssoled[0], 0, 1);
        oledFill(&ssoled[1], 0, 1);
        segundosJugador[0]=persist.tiempo;
        segundosJugador[1]=segundosJugador[0];
        bonus = persist.bonus;
        EEPROM.put(0,persist);
        mod = 0;
        moves[0] = 0;
        moves[1] = 0;
    }else if (type==2){
      int mod = 0;
      player = PAUSE;
        oledFill(&ssoled[0], 0, 1);
        oledFill(&ssoled[1], 0, 1);
        oledWriteString(&ssoled[0], 0, 0, 0, (char *)"Current game modif.",
FONT_SMALL, 0, 1);
        oledWriteString(&ssoled[1], 0, 0, 7, (char *)"PAUSE: next/conf.",
FONT_NORMAL, 0, 1);
        oledWriteString(&ssoled[0], 0, 0, 7, (char *)"+/-: modify", FONT_NORMAL,
0, 1);
        oledWriteString(&ssoled[0], 0, 30, 5, (char *)"Player 1", FONT_SMALL, 0,
1);
        oledWriteString(&ssoled[1], 0, 30, 5, (char *)"Player 2", FONT_SMALL, 0,
1);
        ToHMS(segundosJugador[0],buf);
        oledWriteString(&ssoled[0], 0, 0, 3, (char *)buf, FONT_STRETCHED, 0, 1);
        ToHMS(segundosJugador[1],buf);
```

```
        oledWriteString(&ssoled[1], 0, 0, 3, (char *)buf, FONT_STRETCHED, 0, 1);
      while (1){
        delay(50);
        ToHMS(segundosJugador[0],buf);
        oledWriteString(&ssoled[0], 0, 0, 3, (char *)buf, FONT_STRETCHED, 0, 1);
        ToHMS(segundosJugador[1],buf);
        oledWriteString(&ssoled[1], 0, 0, 3, (char *)buf, FONT_STRETCHED, 0, 1);
        if (digitalRead(4)== LOW){
          if (mod == 0){
            segundosJugador[0]=segundosJugador[0]+5;
          } else if (mod == 1){
            segundosJugador[1]=segundosJugador[1]+5;
          }
        }
        if (digitalRead(5)== LOW){
          if (mod == 0){
            mod++;
          } else if (mod == 1){
            break;
          }
          else{
            Serial.println("ERROR: parametro incorrecto");
          }
        }
        if (digitalRead(6)== LOW){
          if (mod == 0){
            segundosJugador[0]=segundosJugador[0]-5;
          } else if (mod == 1){
            segundosJugador[1]=segundosJugador[1]-5;
          }
        }
      }
        oledFill(&ssoled[0], 0, 1);
        oledFill(&ssoled[1], 0, 1);
        mod = 0;
    } else {
        Serial.println("ERROR: parametro incorrecto");
    }
}

// Función para convertir cualquier variable a string
String variableToString(int value) {
    return String(value);
}

String variableToString(unsigned long value) {
    return String(value);
}

// Para otros tipos, puedes agregar más sobrecargas si es necesario

// Función para convertir segundos a hh:mm o mm:ss
void mostrarTiempoRestante(SSOLED *oled, unsigned long segundosRestantes) {
    char buffer[15];
```

```
    int decimasRestantes=segundosRestantes%10;
    segundosRestantes=segundosRestantes/10;
    if (segundosRestantes >= 3600) {  // Si hay más de 1 hora
        int horas = segundosRestantes / 3600;
        int minutos = (segundosRestantes % 3600) / 60;
        int segundos = segundosRestantes % 60;
        sprintf(buffer, "%2d:%02d:%02d", horas, minutos, segundos);  // Mostrar
solo horas y minutos
        oledWriteString(oled, 0, 0, 3, buffer, FONT_STRETCHED, 0, 1);
    }else if (segundosRestantes<16){
        int minutos = (segundosRestantes % 3600) / 60;
        int segundos = segundosRestantes % 60;
        sprintf(buffer, "0:%02d.%d", segundos, decimasRestantes);  // Mostrar
solo horas y minutos
        oledWriteString(oled, 0, 18, 3, buffer, FONT_STRETCHED, 0, 1);
    } else {  // Menos de 1 hora
        int minutos = segundosRestantes / 60;
        int segundos = segundosRestantes % 60;
        sprintf(buffer, "%02d:%02d   ", minutos, segundos);  // Mostrar minutos
y segundos
        oledWriteString(oled, 0, 18, 3, buffer, FONT_STRETCHED, 0, 1);
    }
}

char* ToHMS(unsigned long seg, char *buffer) {
  //char buffer[15];
  if (seg >= 36000) {  // Si hay más de 1 hora
    int horas = seg / 36000;
    int minutos = (seg % 36000) / 600;
    int segundos = seg % 6;
    sprintf(buffer, "%2d:%02d:%02d   ", horas, minutos, segundos);  // Mostrar
solo horas y minutos
    return buffer;
  } else {  // Menos de 1 hora
    int minutos = seg / 600;
    int segundos = seg % 60;
    sprintf(buffer, " %02d:%02d      ", minutos, segundos);  // Mostrar minutos
y segundos
    return buffer;
  }
}

// Función para mostrar "Bonus" en la parte inferior izquierda de la pantalla
void mostrarBonus() {
    char buf[3];
    char text[7]="Bonus ";
    sprintf(bonus,buf);
    if (bonus != 0) {
      oledWriteString(&ssoled[1], 0, 0, 0, "Bonus", FONT_SMALL, 0, 1);
    } else{
      oledWriteString(&ssoled[1], 0, 0, 0, "      ", FONT_SMALL, 0, 1);
    }
}
```

```
// Función para mostrar movimientos
void mostrarMovimientos() {
    if (moves[0]>moves[1]){
      movimientos = moves[0];
    }else if (moves[0]<moves[1]){
      movimientos = moves[1];
    } else {

    }
    oledWriteString(&ssoled[0], 0, 45, 7, (char *)("Moves: " +
variableToString(movimientos)).c_str(), FONT_NORMAL, 0, 1);
}

void refrescaDisplay(SSOLED *pantalla, unsigned long segunds, int mov) {
  char buf[20];
  if (player!=TIMEOUT){
    mostrarTiempoRestante(pantalla, segunds);  // Mostrar tiempo restante
    mostrarBonus();  // Mostrar "Bonus" en la pantalla del Jugador 1
    mostrarMovimientos();  // Mostrar movimientos del Jugador 1
  }
  if (persist.beep==1){
    oledWriteString(&ssoled[0], 0, 0, 0, "         ", FONT_SMALL, 0, 1);
  } if (persist.beep==0){
    oledWriteString(&ssoled[0], 0, 0, 0, "Sound off", FONT_SMALL, 0, 1);
  }
    /*String str=String("Battery:
")+String(2*(analogRead(0)*(5.0/1024.0)))+String("Vdc");
    str.toCharArray(buf,20);
    variableToString((analogRead(0)/5)).toCharArray(buf, 15);
    oledWriteString(&ssoled[0], 0, 0, 1, buf, FONT_SMALL, 0, 1);
    oledWriteString(&ssoled[0], 0, 12, 1, "   ", FONT_SMALL, 0, 1);
    oledWriteString(&ssoled[0], 0, 30, 1, buf, FONT_SMALL, 0, 1);*/
    if (link==true){
      oledWriteString(&ssoled[1], 0, 60, 0, "  ChessLink", FONT_SMALL, 0, 1);
    }
      if ((analogRead(0)+102)/102 < 5.2){
        oledWriteString(&ssoled[0], 0, 60, 0, "        USB", FONT_SMALL, 0, 1);
      } else if ((analogRead(0)+102)/102 < 7.4){
        oledWriteString(&ssoled[0], 0, 60, 0, "Battery low", FONT_SMALL, 0, 1);
      } else{
        oledWriteString(&ssoled[0], 0, 60, 0, "            ", FONT_SMALL, 0, 1);
    }
}

void SerialOvrd(){
  if (link==true){
    Serial.print(segundosJugador[0]);
    Serial.print(",");
    Serial.print(segundosJugador[1]);
    Serial.print(",");
    Serial.println(movimientos);
  }
}
void loop() {
```

```
    char buf[40];
    for (int i = 0; i < 2; i++) {
      refrescaDisplay(&ssoled[i], segundosJugador[i], moves[i]);

    }

    clock = millis();
    //milisegundos = clock;
    if (clock > millisMax){
      digitalWrite(7, LOW);
    }
    if (Serial.available()) {
    // unsigned long currentTime = millis();
    String command = Serial.readStringUntil('\n');  // Read incoming command
      if (command == "LINK LOGON") {
      // Send LOGON-CONFIRM back to the Python side
      Serial.println("LOGON CONFIRM");
      link = true;
    }
    if (command=="DEFAULT"){
      player=PAUSE;
      persist.tiempo=6000;
      segundosJugador[0]=6000;
      segundosJugador[1]=6000;
      persist.bonus=0;
      persist.beep=1;
      EEPROM.put(0, persist);
    } if (command=="BEEP"){
      if (persist.beep==0){
        persist.beep=1;
        for (int i = 0; i < 2; i++) {
          digitalWrite(7, HIGH);
          delay(25);
          digitalWrite(7, LOW);
          delay(25);
        }
      } else if (persist.beep==1){
        persist.beep=0;
      } else {
        Serial.println("ERROR: incorrect value, returning to known state (beep
on)");
        persist.beep=1;
      }
      EEPROM.put(0, persist);
    } if (command=="HELP"){
      Serial.println("DEFAULT: Flash default settings onto the memory (10 min,
no bonus, beep enabled. Useful when slashing onto Arduino or if u broke
something");
      Serial.println("BEEP: Toggles the beeping on or off");
      Serial.println("LINK LOGON: Sends a ChessLink logon request. Will start
sending data through Serial. Rarely used manually.");
      Serial.println("HELP: Shows this prompt.");
    } else{
      Serial.println("ERROR: "+command+" is not a valid command");
```

```
    }
    int player1Time = segundosJugador[0]; // Replace with actual logic
    int player2Time = segundosJugador[1];

    // Send formatted data: player1_time,player2_time

  }
    if (clock > (ultimoclock + 99)) {
    if (link==true){
      Serial.print(segundosJugador[0]);
      Serial.print(",");
      Serial.print(segundosJugador[1]);
      Serial.print(",");
      Serial.println(movimientos);
      }
        ultimoclock = clock;

        switch (player) {
        case 0:

            if (segundosJugador[0] > 0) {
                segundosJugador[0]--;
            }
            if (segundosJugador[0] == 0) {
              player=TIMEOUT;
              beep(1500);
              digitalWrite(9, HIGH);
              //ToHMS(segundosJugador[1], buf);
              sprintf(buf, ">0:00.0<");
              oledWriteString(&ssoled[0], 0, 0, 3, (char *)buf, FONT_STRETCHED,
0, 1);
            }
            if (segundosJugador[0] == 30 || segundosJugador[0] == 20 ||
segundosJugador[0] == 10) {
              beep(100);
            }
            break;

        case 1:
            if (segundosJugador[1] > 0) {
                segundosJugador[1]--;
            }
            if (segundosJugador[1] == 0) {
              player=TIMEOUT;
              beep(1500);
              digitalWrite(10, HIGH);
              //ToHMS(segundosJugador[1], buf);
              sprintf(buf, ">0:00.0<");
              oledWriteString(&ssoled[1], 0, 0, 3, (char *)buf, FONT_STRETCHED,
0, 1);
            }
            if (segundosJugador[1] == 30 || segundosJugador[1] == 20 ||
segundosJugador[1] == 10) {
              beep(100);
```

```
            }
            break;

        case PAUSE:
        oledWriteString(&ssoled[0], 0, 0, 6, (char *)" ", FONT_STRETCHED, 0, 1);
        oledWriteString(&ssoled[1], 0, 110, 6, (char *)" ", FONT_STRETCHED, 0,
1);
            break;

        case TIMEOUT:
            break;

        default:
            oledWriteString(&ssoled[0], 0, 60, 7, (char *)"ERROR", FONT_NORMAL,
0, 1);
            Serial.println("Error: estado desconocido");
        }


    }

    // Detectar el botón del Jugador 1
    if ((digitalRead(2) == LOW) && !botonPresionado[0] && (player == 0 || player
== PAUSE)) {
      SerialOvrd();
        if(player!=PAUSE){
            segundosJugador[0] += bonus;  // Agregar bonus al Jugador 2
            moves[0]++;  // Incrementar movimientos del Jugador 1
        }
            oledWriteString(&ssoled[0], 0, 0, 6, (char *)" ", FONT_STRETCHED, 0,
1);
            oledWriteString(&ssoled[1], 0, 110, 6, (char *)">", FONT_STRETCHED,
0, 1);
        botonPresionado[0] = true;
        //Serial.println("Cambio a Jugador 2");
        player = 1;  // Cambia al Jugador 2
     } else {
        botonPresionado[0] = false;
    }

    // Detectar el botón del Jugador 2
    if ((digitalRead(3) == LOW) && !botonPresionado[1] && (player == 1 || player
== PAUSE)) {
      SerialOvrd();
      if(player!=PAUSE){
        segundosJugador[1] += bonus;
        moves[1]++;  // Incrementar movimientos del Jugador 1
      }
        player = 0;  // Cambia al Jugador 1
        //segundosJugador[1] += bonus;  // Agregar bonus al Jugador 1
        botonPresionado[1] = true;
        //Serial.println("Cambio a Jugador 1");
            oledWriteString(&ssoled[0], 0, 0, 6, (char *)"<", FONT_STRETCHED, 0,
1);
```

```
            oledWriteString(&ssoled[1], 0, 110, 6, (char *)" ", FONT_STRETCHED,
0, 1);
    } else {
        botonPresionado[1] = false;
    }


    // Detectar el botón del PAUSE
    if (digitalRead(5) == LOW && digitalRead(2) == HIGH && digitalRead(3) ==
HIGH) {
      player = PAUSE;
        oledWriteString(&ssoled[0], 0, 0, 6, (char *)" ", FONT_STRETCHED, 0, 1);
        oledWriteString(&ssoled[1], 0, 110, 6, (char *)" ", FONT_STRETCHED, 0,
1);
    }

    if (digitalRead(4) == LOW && player==PAUSE && digitalRead(2) == HIGH) {
      timeSetting(1);
    }
    if (digitalRead(6) == LOW && player==PAUSE && digitalRead(3) == HIGH) {
      timeSetting(2);
    }
}
```