#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

// Oppdater I2C-adressen med den du fant (for eksempel 0x27)

LiquidCrystal\_I2C lcd(0x27, 16, 2); // Adresse, kolonner, rader

// Definer pinnene for hjemmelag, bortelag, pause og minusmål

const int H\_PIN = 11; // Hjemmelag

const int B\_PIN = 7; // Bortelag

const int PAUSE\_PIN = 9; // Pause tid

const int MINUS\_PLAYER\_PIN = 12; // Minus mål for spiller

const int MINUS\_BOT\_PIN = 6; // Minus mål for bot

// Variabler for å lagre antall mål

int hjemmelag\_mal = 0;

int bortelag\_mal = 0;

bool paused = false; // Variabel som angir om tiden er pauset

// Debounce-relaterte variabler

unsigned long lastDebounceTimeH = 0; // Tid for siste endring på H\_PIN

unsigned long lastDebounceTimeB = 0; // Tid for siste endring på B\_PIN

unsigned long lastDebounceTimePause = 0; // Debounce for pause

unsigned long lastDebounceTimeMinusPlayer = 0; // Debounce for minus mål for spiller

unsigned long lastDebounceTimeMinusBot = 0; // Debounce for minus mål for bot

const unsigned long debounceDelay = 500; // 500ms debounce delay

// Variabler for tid

unsigned long startTid;

unsigned long elapsedTid = 0; // Lagret tid når pausen starter

int displayMinutt = 0;

int displaySekund = 0;

unsigned long pausedTid = 0; // Lagret tid da kampen ble pauset

// Tidsakselerasjonsfaktor

const float tidsFaktor = 4.5; // 45 minutter på 10 minutter (45/10 = 4.5)

void setup() {

// Initialiser LCD-skjermen

lcd.init();

lcd.backlight();

// Sett opp pin 2, 3, 4, 5 og 6 som digitale innganger med interne pull-up-motstander

pinMode(H\_PIN, INPUT\_PULLUP);

pinMode(B\_PIN, INPUT\_PULLUP);

pinMode(PAUSE\_PIN, INPUT\_PULLUP);

pinMode(MINUS\_PLAYER\_PIN, INPUT\_PULLUP);

pinMode(MINUS\_BOT\_PIN, INPUT\_PULLUP);

// Start tidtakeren

startTid = millis();

// Vis initial informasjon på LCD-skjermen

oppdaterSkjerm();

}

void loop() {

// Les statusen på H\_PIN og B\_PIN

int hState = digitalRead(H\_PIN);

int bState = digitalRead(B\_PIN);

int pauseState = digitalRead(PAUSE\_PIN);

int minusPlayerState = digitalRead(MINUS\_PLAYER\_PIN);

int minusBotState = digitalRead(MINUS\_BOT\_PIN);

// Håndter mål for hjemmelaget (Player)

if (hState == HIGH && (millis() - lastDebounceTimeH) > debounceDelay) { // Sjekker om knappen er trykket (LOW)

hjemmelag\_mal++; // Øk mål for hjemmelaget

oppdaterSkjerm(); // Oppdater skjermen for å vise målet

lastDebounceTimeH = millis(); // Oppdater debounce-tid for hjemmelaget

}

// Håndter mål for bortelaget (Bot)

if (bState == HIGH && (millis() - lastDebounceTimeB) > debounceDelay) { // Sjekker om knappen er trykket (LOW)

bortelag\_mal++; // Øk mål for bortelaget

oppdaterSkjerm(); // Oppdater skjermen for å vise målet

lastDebounceTimeB = millis(); // Oppdater debounce-tid for bortelaget

}

// Pauser eller starter kampen

if (pauseState == HIGH && (millis() - lastDebounceTimePause) > debounceDelay) { // Sjekker om pauseknappen er trykket (LOW)

paused = !paused; // Bytter mellom pause og start

if (paused) {

// Lagre tiden som har gått når vi pauser

pausedTid = millis();

} else {

// Fortsett tiden der vi sluttet ved å justere starttiden

startTid += (millis() - pausedTid);

}

lastDebounceTimePause = millis();

}

// Minus mål for spiller (Player)

if (minusPlayerState == HIGH && (millis() - lastDebounceTimeMinusPlayer) > debounceDelay) { // Sjekker om knappen er trykket (LOW)

if (hjemmelag\_mal > 0) hjemmelag\_mal--; // Reduser mål for hjemmelaget

oppdaterSkjerm();

lastDebounceTimeMinusPlayer = millis();

}

// Minus mål for bot (Bot)

if (minusBotState == HIGH && (millis() - lastDebounceTimeMinusBot) > debounceDelay) { // Sjekker om knappen er trykket (LOW)

if (bortelag\_mal > 0) bortelag\_mal--; // Reduser mål for bortelaget

oppdaterSkjerm();

lastDebounceTimeMinusBot = millis();

}

// Oppdater skjerm og tid hvis spillet ikke er pauset

if (!paused) {

// Beregn forløpt tid med tidsakselerasjon

elapsedTid = millis() - startTid;

unsigned long akselerertTid = elapsedTid \* tidsFaktor;

// Konverter til minutter og sekunder

displayMinutt = (akselerertTid / 60000) % 60;

displaySekund = (akselerertTid / 1000) % 60;

// Oppdater skjermen hvert sekund

static unsigned long lastUpdate = 0;

if (millis() - lastUpdate >= 1000) {

lastUpdate = millis();

oppdaterSkjerm();

}

}

// Stopp når tiden når 45:00

if (displayMinutt >= 45 && !paused) {

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Kampen er over");

lcd.setCursor(0, 1);

lcd.print("H:" + String(hjemmelag\_mal) + " B:" + String(bortelag\_mal));

while (true)

;

}

}

// Funksjon for å oppdatere LCD-skjermen

void oppdaterSkjerm() {

lcd.clear();

// Øverste rad: --YOU--TIME--BOT--

lcd.setCursor(0, 0);

lcd.print("SPILLER TID BOT");

// Formater minutter og sekunder til MM:SS

char tidBuffer[6];

sprintf(tidBuffer, "%02d:%02d", displayMinutt, displaySekund);

// Nederste rad: ---XX---MM:SS--XX--

lcd.setCursor(0, 1);

lcd.print(" ");

lcd.print(hjemmelag\_mal); // Hjemmelagets mål

lcd.print(" ");

lcd.print(tidBuffer); // Tid i formatet MM:SS

lcd.print(" ");

lcd.print(bortelag\_mal); // Bortelagets mål

lcd.print(" ");

}