

**Criação do Código**

**João Vitor Morais Souza.**

**São José dos Campos - SP**

**2025**

1. Introdução.

O presente documento tem como objetivo descrever o processo de desenvolvimento do código utilizado no projeto de um placar eletrônico, incluindo funcionalidades como controle de tempo, pontuação e comunicação via Bluetooth. O projeto foi realizado de forma progressiva, com apoio de pesquisas online, vídeos e auxílio da inteligência artificial ChatGPT.

2. Desenvolvimento do Código.

# **2.1 Cronômetro.** O primeiro passo para a criação do código foi o desenvolvimento do cronômetro, exibido em um display de 4 dígitos do tipo TM1637. Inicialmente, foi utilizado um código-fonte encontrado em vídeos no YouTube, que foi adaptado e melhorado com auxílio do ChatGPT. As modificações incluíram ajustes na lógica de contagem, nos botões de controle e na exibição precisa do tempo.

O sistema permite iniciar e parar o tempo,acréscimo de um minuto e reset do tempo por meio dos botões dedicados

.

# **2.2 Pontuação.** Em seguida, foi criado o código responsável pelo controle de pontuação. A base desse código foi um projeto anterior do semestre passado, que foi reaproveitado e reestruturado para se adequar à nova necessidade. Com ajuda do ChatGPT, o código foi expandido para controlar três pares de displays duplos (dois dígitos para cada item), representando:

* Pontuação do Time 1
* Pontuação do Time 2
* Rodada atual (Round)

O sistema permite tanto o acréscimo quanto a remoção de pontos por meio de botões dedicados a cada display.

# **2.3 Código Final Integrado.** Após o desenvolvimento individual de cada módulo (cronômetro e pontuação), os dois códigos foram unificados em um código final. Essa junção exigiu ajustes de variáveis, sincronização de timers e gerenciamento das entradas dos botões para garantir o funcionamento conjunto e estável do sistema completo.

**3. Comunicação via Bluetooth (Teoria).**

A pedido dos professores, foi desenvolvida uma proposta para permitir o controle do placar via comunicação Bluetooth. Essa etapa do projeto foi realizada de forma teórica, ou seja, o sistema não foi testado na prática, mas todo o planejamento e desenvolvimento do código foram concluídos.

Primeiramente, foram realizadas pesquisas online e assistidos vídeos explicativos para entender o funcionamento da comunicação Bluetooth com Arduino, principalmente com o uso do módulo HC-05. A partir dessas informações, foi desenvolvido o código necessário para que o Arduino recebesse comandos enviados por Bluetooth, como iniciar/parar o cronômetro, aumentar/diminuir os pontos e mudar a rodada.

Além disso, desenvolvi um aplicativo próprio para celular com o objetivo de enviar os comandos ao placar eletrônico de forma remota. A criação do aplicativo foi feita com base em tutoriais disponíveis no YouTube, adaptando o conteúdo apresentado para atender às funcionalidades específicas do meu projeto. O aplicativo oferece uma interface simples e eficiente, facilitando o controle das principais funções do placar.

Embora essa etapa ainda não tenha sido testada fisicamente, todo o código e estrutura lógica já estão prontos para implementação futura.

**4. Conclusão.** O desenvolvimento do código do placar eletrônico envolveu a combinação de diferentes fontes de aprendizado: experiências anteriores, tutoriais online e apoio de inteligência artificial. O projeto proporcionou a aplicação prática de conhecimentos de programação, eletrônica e integração de hardware, além de exigir organização e adaptação para implementação de novas funcionalidades, como o controle via Bluetooth.

**5. Fonte de pesquisa.**

# **5.1 Referência sobre o display de 4 dígitos.**

**Cronômetro Arduino com Display 7 Segmentos TM1637**

<https://www.usinainfo.com.br/blog/cronometro-arduino-com-display-7-segmentos-tm1637/?srsltid=AfmBOoqtZKV4IvnJfbp7u4QA17StWZP28qGTKflo6-SFwX7QAUXujKEf>

# **Projeto 64 - Cronômetro com display TM1637 e Arduino**

<https://www.squids.com.br/arduino/projetos-arduino/projetos-squids/basico/228-projeto-64-cronometro-com-display-tm1637-e-arduino>

**Mamute Eletrônica.** 120 - Módulo Display TM1637.

<https://www.youtube.com/watch?v=NqkBAA67KMg>

**FunBots.** Como usar um DISPLAY DE 7 SEGMENTOS com 4 Dígitos | Multiplexando com Arduino.

<https://www.youtube.com/watch?v=XrLXFoIkC8E>

**Angelo Luis Ferreira.** Usando o módulo TM1637 (display 4 dígitos) com Arduino - #63.

<https://www.youtube.com/watch?v=Y6E323UrOI4>

**Arduino Maker.** COMO UTILIZAR DISPLAY DE 7 SEGMENTOS E 4 DÍGITOS COM ARDUINO[Vamos fazer um relógio digital].

<https://www.youtube.com/watch?v=0aClykRoFkw>

# **5.2 Referência sobre o Comunicação via Bluetooth.**

# **Como utilizar o módulo Bluetooth HC-05 com Arduino.**

# [**https://www.blogdarobotica.com/2023/02/13/como-utilizar-o-modulo-bluetooth-hc-05-com-arduino/**](https://www.blogdarobotica.com/2023/02/13/como-utilizar-o-modulo-bluetooth-hc-05-com-arduino/)

# 

# 

# **Tutorial Módulo Bluetooth com Arduino.**

# [**https://www.makerhero.com/blog/tutorial-modulo-bluetooth-com-arduino/?srsltid=AfmBOopG\_9mgioXTVD0fW5GRfRxOMaX-2nJgKUjRIPA\_f-x8XoTwQfnT**](https://www.makerhero.com/blog/tutorial-modulo-bluetooth-com-arduino/?srsltid=AfmBOopG_9mgioXTVD0fW5GRfRxOMaX-2nJgKUjRIPA_f-x8XoTwQfnT)

# **Brincando com Ideias.** Aprenda a Usar o Módulo Bluetooth HC-06 com Arduino

<https://www.youtube.com/watch?v=8AoFQc2J9xI&list=LL&index=12&t=1115s>

# **FunBots.** APP GRATUITO BLUETOOTH PARA ARDUINO | Aprenda fácil! Dabble Bluetooth Arduino com HC-05 e HM-10.

<https://www.youtube.com/watch?v=f4wSzpJJlsk>

**Professor Douglas Maioli.** Como criar seu primeiro aplicativo no APP Inventor.

<https://www.youtube.com/watch?v=3wM_0PPYDGU>

**Brino Robótica Educacional.** CRIE SEU PRIMEIRO APP | Tutorial Passo a Passo

<https://www.youtube.com/watch?v=B9KFH7x3RWk>

**6. Códigos desenvolvidos.**

# **6.1 Código antigo.**

#include <ShiftRegister74HC595.h>

// Pinos do 74HC595

#define SDI2 10

#define SCLK2 9

#define LOAD2 8

#define SDI3 7

#define SCLK3 6

#define LOAD3 5

// Pinos dos botões

#define BUTTON\_PIN\_START\_STOP 0 // Botão de iniciar/parar

#define BUTTON\_PIN\_RESET 1 // Botão de reset

#define BUTTON\_EXTRA\_TIME 2 // Botão de tempo extra

#define BUTTON\_PIN\_INCREMENT\_ROUND 3 // Botão para incrementar o round

#define BUTTON\_PIN\_DECREMENT\_ROUND 4 // Botão para decrementar o round

int currentRound = 1; // Inicializa o round em 1

bool isRunning = false;

bool extraTimeEnabled = false; // Variável que indica se o tempo extra foi ativado

unsigned long startTime = 0;

unsigned long pausedTime = 0;

int displayTime = 0; // Variável para exibir o tempo no display

ShiftRegister74HC595<2> sr1(SDI2, SCLK2, LOAD2);

ShiftRegister74HC595<2> sr3(SDI3, SCLK3, LOAD3);

// Padrões para os dígitos de 0 a 9 no display de 7 segmentos

uint8\_t digits[] = {

B11000000, // 0

B11111001, // 1

B10100100, // 2

B10110000, // 3

B10011001, // 4

B10010010, // 5

B10000010, // 6

B11111000, // 7

B10000000, // 8

B10010000 // 9

};

void showNumber(ShiftRegister74HC595<2>& sr, int num) {

int digit2 = num % 10;

int digit1 = (num / 10) % 10;

uint8\_t numberToPrint[] = {digits[digit1], digits[digit2]};

sr.setAll(numberToPrint);

}

void setup() {

pinMode(BUTTON\_PIN\_START\_STOP, INPUT\_PULLUP);

pinMode(BUTTON\_PIN\_RESET, INPUT\_PULLUP);

pinMode(BUTTON\_EXTRA\_TIME, INPUT\_PULLUP);

pinMode(BUTTON\_PIN\_INCREMENT\_ROUND, INPUT\_PULLUP);

pinMode(BUTTON\_PIN\_DECREMENT\_ROUND, INPUT\_PULLUP);

// Inicializa o Round 1 no display

showNumber(sr1, currentRound);

showNumber(sr3, 0); // Inicializa o display de tempo com zero

}

void loop() {

// Botão de iniciar/parar

if (digitalRead(BUTTON\_PIN\_START\_STOP) == LOW) {

delay(50); // Antirrebote

if (digitalRead(BUTTON\_PIN\_START\_STOP) == LOW) {

isRunning = !isRunning;

if (isRunning) {

startTime = millis() - pausedTime; // Retorna ao tempo de onde parou

} else {

pausedTime = millis() - startTime; // Salva o tempo pausado

}

while (digitalRead(BUTTON\_PIN\_START\_STOP) == LOW); // Espera soltar o botão

}

}

// Botão de reset

if (digitalRead(BUTTON\_PIN\_RESET) == LOW) {

delay(50); // Antirrebote

if (digitalRead(BUTTON\_PIN\_RESET) == LOW) {

isRunning = false;

pausedTime = 0;

startTime = 0;

displayTime = 0;

extraTimeEnabled = false;

showNumber(sr3, displayTime);

while (digitalRead(BUTTON\_PIN\_RESET) == LOW); // Espera soltar o botão

}

}

// Botão de tempo extra

if (digitalRead(BUTTON\_EXTRA\_TIME) == LOW) {

delay(50); // Antirrebote

if (digitalRead(BUTTON\_EXTRA\_TIME) == LOW) {

// Só permite adicionar tempo quando exatamente em 60 segundos e parado

if (displayTime == 60 && !extraTimeEnabled && !isRunning) {

extraTimeEnabled = true;

startTime = millis() - 60000; // Mantém o tempo em 60 segundos

pausedTime = 60000;

isRunning = true;

while (digitalRead(BUTTON\_EXTRA\_TIME) == LOW); // Espera soltar o botão

}

}

}

// Botão para incrementar o round

if (digitalRead(BUTTON\_PIN\_INCREMENT\_ROUND) == LOW) {

delay(50); // Antirrebote

if (digitalRead(BUTTON\_PIN\_INCREMENT\_ROUND) == LOW) {

if (currentRound < 4) {

currentRound++;

}

showNumber(sr1, currentRound);

while (digitalRead(BUTTON\_PIN\_INCREMENT\_ROUND) == LOW); // Espera soltar o botão

}

}

// Botão para decrementar o round

if (digitalRead(BUTTON\_PIN\_DECREMENT\_ROUND) == LOW) {

delay(50); // Antirrebote

if (digitalRead(BUTTON\_PIN\_DECREMENT\_ROUND) == LOW) {

if (currentRound > 1) {

currentRound--;

}

showNumber(sr1, currentRound);

while (digitalRead(BUTTON\_PIN\_DECREMENT\_ROUND) == LOW); // Espera soltar o botão

}

}

// Controle do temporizador

if (isRunning) {

unsigned long currentTime = millis();

displayTime = (currentTime - startTime) / 1000; // Converte para segundos

// Verifica se atingiu 60 segundos sem tempo extra ativado

if (displayTime >= 60 && !extraTimeEnabled) {

displayTime = 60;

isRunning = false;

}

// Verifica se atingiu 90 segundos com tempo extra ativado

else if (displayTime >= 90 && extraTimeEnabled) {

displayTime = 90;

isRunning = false;

}

// Atualiza o display com o tempo atual

showNumber(sr3, displayTime);

}

}

# **6.2 Código do cronômetro.**

#include <TM1637Display.h>

#define CLK 2 // Pino do relógio do display

#define DIO 3 // Pino de dados do display

#define BTN\_START\_STOP 4 // Botão iniciar/parar

#define BTN\_ADD\_TIME 5 // Botão acrescentar tempo

#define BTN\_RESET 6 // Botão zerar

TM1637Display display(CLK, DIO);

unsigned long previousMillis = 0;

bool running = false;

int defaultMaxTime = 120; // Tempo máximo inicial (2 minutos)

int maxTime = 120;

int seconds = 0;

void setup() {

pinMode(BTN\_START\_STOP, INPUT\_PULLUP);

pinMode(BTN\_ADD\_TIME, INPUT\_PULLUP);

pinMode(BTN\_RESET, INPUT\_PULLUP);

display.setBrightness(7);

display.showNumberDecEx(0, 0x40, true); // Exibe 00:00

}

void loop() {

// Botão Iniciar/Parar

if (digitalRead(BTN\_START\_STOP) == LOW) {

delay(200);

running = !running;

while (digitalRead(BTN\_START\_STOP) == LOW); // Aguarda o botão ser solto

}

// Botão Acrescentar Tempo (Permite acréscimo mesmo após o cronômetro parar e reinicia se já parou)

if (digitalRead(BTN\_ADD\_TIME) == LOW) {

delay(200);

if (maxTime < 180) {

maxTime += 60; // Acrescenta 1 minuto até o limite de 3 minutos

if (!running && seconds >= maxTime - 60) {

running = true; // Reinicia a contagem se já tiver parado

}

}

while (digitalRead(BTN\_ADD\_TIME) == LOW); // Aguarda o botão ser solto

}

// Botão Resetar (Volta ao tempo padrão de 2 minutos)

if (digitalRead(BTN\_RESET) == LOW) {

delay(200);

running = false;

seconds = 0;

maxTime = defaultMaxTime; // Reseta o tempo máximo para 2 minutos

while (digitalRead(BTN\_RESET) == LOW); // Aguarda o botão ser solto

}

// Atualiza o tempo

if (running && millis() - previousMillis >= 1000) {

previousMillis = millis();

if (seconds < maxTime) {

seconds++;

} else {

running = false; // Para automaticamente ao atingir o limite

}

}

displayTime(seconds);

}

void displayTime(int sec) {

int minutes = sec / 60;

int secRemain = sec % 60;

int timeToDisplay = (minutes \* 100) + secRemain;

display.showNumberDecEx(timeToDisplay, 0x40, true);

}

# **6.3 Código dos pontos.**

#include <ShiftRegister74HC595.h>

// Pinos do 74HC595 para o display do round

#define SDI2 10

#define SCLK2 9

#define LOAD2 8

// Pinos dos botões

#define BUTTON\_PIN\_INCREMENT\_ROUND 3 // Botão para incrementar o round

#define BUTTON\_PIN\_DECREMENT\_ROUND 4 // Botão para decrementar o round

// Inicializa o registrador para 2 dígitos (2 x 7 segmentos)

ShiftRegister74HC595<2> sr1(SDI2, SCLK2, LOAD2);

// Vetor com os padrões dos dígitos 0 a 9 para display de 7 segmentos

uint8\_t digits[] = {

B11000000, // 0

B11111001, // 1

B10100100, // 2

B10110000, // 3

B10011001, // 4

B10010010, // 5

B10000010, // 6

B11111000, // 7

B10000000, // 8

B10010000 // 9

};

int currentRound = 1; // Round começa em 1

// Função para mostrar número de 2 dígitos no display

void showNumber(ShiftRegister74HC595<2>& sr, int num) {

int digit2 = num % 10; // unidades

int digit1 = (num / 10) % 10; // dezenas

uint8\_t numberToPrint[] = {digits[digit1], digits[digit2]};

sr.setAll(numberToPrint);

}

void setup() {

pinMode(BUTTON\_PIN\_INCREMENT\_ROUND, INPUT\_PULLUP);

pinMode(BUTTON\_PIN\_DECREMENT\_ROUND, INPUT\_PULLUP);

showNumber(sr1, currentRound); // Exibe o round inicial

}

void loop() {

// Botão para incrementar o round

if (digitalRead(BUTTON\_PIN\_INCREMENT\_ROUND) == LOW) {

delay(50); // antirrebote

if (digitalRead(BUTTON\_PIN\_INCREMENT\_ROUND) == LOW) {

if (currentRound < 4) {

currentRound++;

}

showNumber(sr1, currentRound);

while (digitalRead(BUTTON\_PIN\_INCREMENT\_ROUND) == LOW); // Espera soltar

}

}

// Botão para decrementar o round

if (digitalRead(BUTTON\_PIN\_DECREMENT\_ROUND) == LOW) {

delay(50); // antirrebote

if (digitalRead(BUTTON\_PIN\_DECREMENT\_ROUND) == LOW) {

if (currentRound > 1) {

currentRound--;

}

showNumber(sr1, currentRound);

while (digitalRead(BUTTON\_PIN\_DECREMENT\_ROUND) == LOW); // Espera soltar

}

}

}

# **6.4 Código dos pontos X3.**

#include <ShiftRegister74HC595.h>

// ==== Pinos do 74HC595 para cada grupo de displays ====

#define SDI\_ROUND 10

#define SCLK\_ROUND 9

#define LOAD\_ROUND 8

#define SDI\_TIME1 7

#define SCLK\_TIME1 6

#define LOAD\_TIME1 5

#define SDI\_TIME2 13

#define SCLK\_TIME2 12

#define LOAD\_TIME2 11

// ==== Botões Round ====

#define BTN\_ROUND\_INC 2

#define BTN\_ROUND\_DEC 3

// ==== Botões Time 1 ====

#define BTN\_TIME1\_INC 4

#define BTN\_TIME1\_DEC 5

// ==== Botões Time 2 ====

#define BTN\_TIME2\_INC 6

#define BTN\_TIME2\_DEC 7

// ==== Instâncias dos registradores de deslocamento ====

ShiftRegister74HC595<2> srRound(SDI\_ROUND, SCLK\_ROUND, LOAD\_ROUND);

ShiftRegister74HC595<2> srTime1(SDI\_TIME1, SCLK\_TIME1, LOAD\_TIME1);

ShiftRegister74HC595<2> srTime2(SDI\_TIME2, SCLK\_TIME2, LOAD\_TIME2);

// ==== Tabela de padrões para os dígitos 0 a 9 ====

uint8\_t digits[] = {

B11000000, // 0

B11111001, // 1

B10100100, // 2

B10110000, // 3

B10011001, // 4

B10010010, // 5

B10000010, // 6

B11111000, // 7

B10000000, // 8

B10010000 // 9

};

// ==== Variáveis ====

int roundValue = 0;

int time1Value = 0;

int time2Value = 0;

// ==== Função para exibir números nos displays ====

void showNumber(ShiftRegister74HC595<2>& sr, int num) {

int unidades = num % 10;

int dezenas = (num / 10) % 10;

uint8\_t toPrint[] = {digits[dezenas], digits[unidades]};

sr.setAll(toPrint);

}

// ==== Setup ====

void setup() {

// Configura pinos dos botões como entradas com pull-up

pinMode(BTN\_ROUND\_INC, INPUT\_PULLUP);

pinMode(BTN\_ROUND\_DEC, INPUT\_PULLUP);

pinMode(BTN\_TIME1\_INC, INPUT\_PULLUP);

pinMode(BTN\_TIME1\_DEC, INPUT\_PULLUP);

pinMode(BTN\_TIME2\_INC, INPUT\_PULLUP);

pinMode(BTN\_TIME2\_DEC, INPUT\_PULLUP);

// Exibe valores iniciais

showNumber(srRound, roundValue);

showNumber(srTime1, time1Value);

showNumber(srTime2, time2Value);

}

// ==== Loop principal ====

void loop() {

// --- Controle do Round ---

if (digitalRead(BTN\_ROUND\_INC) == LOW) {

delay(50);

if (digitalRead(BTN\_ROUND\_INC) == LOW && roundValue < 99) {

roundValue++;

showNumber(srRound, roundValue);

while (digitalRead(BTN\_ROUND\_INC) == LOW);

}

}

if (digitalRead(BTN\_ROUND\_DEC) == LOW) {

delay(50);

if (digitalRead(BTN\_ROUND\_DEC) == LOW && roundValue > 0) {

roundValue--;

showNumber(srRound, roundValue);

while (digitalRead(BTN\_ROUND\_DEC) == LOW);

}

}

// --- Controle do Time 1 ---

if (digitalRead(BTN\_TIME1\_INC) == LOW) {

delay(50);

if (digitalRead(BTN\_TIME1\_INC) == LOW && time1Value < 99) {

time1Value++;

showNumber(srTime1, time1Value);

while (digitalRead(BTN\_TIME1\_INC) == LOW);

}

}

if (digitalRead(BTN\_TIME1\_DEC) == LOW) {

delay(50);

if (digitalRead(BTN\_TIME1\_DEC) == LOW && time1Value > 0) {

time1Value--;

showNumber(srTime1, time1Value);

while (digitalRead(BTN\_TIME1\_DEC) == LOW);

}

}

// --- Controle do Time 2 ---

if (digitalRead(BTN\_TIME2\_INC) == LOW) {

delay(50);

if (digitalRead(BTN\_TIME2\_INC) == LOW && time2Value < 99) {

time2Value++;

showNumber(srTime2, time2Value);

while (digitalRead(BTN\_TIME2\_INC) == LOW);

}

}

if (digitalRead(BTN\_TIME2\_DEC) == LOW) {

delay(50);

if (digitalRead(BTN\_TIME2\_DEC) == LOW && time2Value > 0) {

time2Value--;

showNumber(srTime2, time2Value);

while (digitalRead(BTN\_TIME2\_DEC) == LOW);

}

}

}

# **6.5 Código Final.**

#include <ShiftRegister74HC595.h>

#include <TM1637Display.h>

// ==== Pinos do 74HC595 ====

#define SDI\_ROUND 39

#define SCLK\_ROUND 41

#define LOAD\_ROUND 43

#define SDI\_TIME1 48

#define SCLK\_TIME1 50

#define LOAD\_TIME1 52

#define SDI\_TIME2 8

#define SCLK\_TIME2 9

#define LOAD\_TIME2 10

// ==== Pinos do TM1637 (TROCADO PARA EVITAR CONFLITO) ====

#define CLK 2

#define DIO 3

// ==== Botões Round ====

#define BTN\_ROUND\_INC 22

#define BTN\_ROUND\_DEC 23

// ==== Botões Time 1 ====

#define BTN\_TIME1\_INC 26

#define BTN\_TIME1\_DEC 27

// ==== Botões Time 2 ====

#define BTN\_TIME2\_INC 30

#define BTN\_TIME2\_DEC 31

// ==== Botões do Cronômetro ====

#define BTN\_START\_STOP 35

#define BTN\_ADD\_TIME 34

#define BTN\_RESET 44

// ==== Instâncias ====

ShiftRegister74HC595<2> srRound(SDI\_ROUND, SCLK\_ROUND, LOAD\_ROUND);

ShiftRegister74HC595<2> srTime1(SDI\_TIME1, SCLK\_TIME1, LOAD\_TIME1);

ShiftRegister74HC595<2> srTime2(SDI\_TIME2, SCLK\_TIME2, LOAD\_TIME2);

TM1637Display display(CLK, DIO);

// ==== Dígitos 7 segmentos ====

uint8\_t digits[] = {

B11000000, B11111001, B10100100, B10110000, B10011001,

B10010010, B10000010, B11111000, B10000000, B10010000

};

// ==== Variáveis ====

int roundValue = 0;

int time1Value = 0;

int time2Value = 0;

unsigned long previousMillis = 0;

bool running = false;

int defaultMaxTime = 120;

int maxTime = 120;

int seconds = 0;

// ==== Funções ====

void showNumber(ShiftRegister74HC595<2>& sr, int num) {

int unidades = num % 10;

int dezenas = (num / 10) % 10;

uint8\_t toPrint[] = {digits[dezenas], digits[unidades]};

sr.setAll(toPrint);

}

void displayTime(int sec) {

int minutes = sec / 60;

int secRemain = sec % 60;

int timeToDisplay = (minutes \* 100) + secRemain;

display.showNumberDecEx(timeToDisplay, 0x40, true);

}

// ==== Setup ====

void setup() {

pinMode(BTN\_ROUND\_INC, INPUT\_PULLUP);

pinMode(BTN\_ROUND\_DEC, INPUT\_PULLUP);

pinMode(BTN\_TIME1\_INC, INPUT\_PULLUP);

pinMode(BTN\_TIME1\_DEC, INPUT\_PULLUP);

pinMode(BTN\_TIME2\_INC, INPUT\_PULLUP);

pinMode(BTN\_TIME2\_DEC, INPUT\_PULLUP);

pinMode(BTN\_START\_STOP, INPUT\_PULLUP);

pinMode(BTN\_ADD\_TIME, INPUT\_PULLUP);

pinMode(BTN\_RESET, INPUT\_PULLUP);

display.setBrightness(7);

display.showNumberDecEx(0, 0x40, true);

showNumber(srRound, roundValue);

showNumber(srTime1, time1Value);

showNumber(srTime2, time2Value);

}

// ==== Loop ====

void loop() {

// Controle Round

if (digitalRead(BTN\_ROUND\_INC) == LOW) {

delay(50);

if (digitalRead(BTN\_ROUND\_INC) == LOW && roundValue < 99) {

roundValue++;

showNumber(srRound, roundValue);

while (digitalRead(BTN\_ROUND\_INC) == LOW);

}

}

if (digitalRead(BTN\_ROUND\_DEC) == LOW) {

delay(50);

if (digitalRead(BTN\_ROUND\_DEC) == LOW && roundValue > 0) {

roundValue--;

showNumber(srRound, roundValue);

while (digitalRead(BTN\_ROUND\_DEC) == LOW);

}

}

// Controle Time 1

if (digitalRead(BTN\_TIME1\_INC) == LOW) {

delay(50);

if (digitalRead(BTN\_TIME1\_INC) == LOW && time1Value < 99) {

time1Value++;

showNumber(srTime1, time1Value);

while (digitalRead(BTN\_TIME1\_INC) == LOW);

}

}

if (digitalRead(BTN\_TIME1\_DEC) == LOW) {

delay(50);

if (digitalRead(BTN\_TIME1\_DEC) == LOW && time1Value > 0) {

time1Value--;

showNumber(srTime1, time1Value);

while (digitalRead(BTN\_TIME1\_DEC) == LOW);

}

}

// Controle Time 2

if (digitalRead(BTN\_TIME2\_INC) == LOW) {

delay(50);

if (digitalRead(BTN\_TIME2\_INC) == LOW && time2Value < 99) {

time2Value++;

showNumber(srTime2, time2Value);

while (digitalRead(BTN\_TIME2\_INC) == LOW);

}

}

if (digitalRead(BTN\_TIME2\_DEC) == LOW) {

delay(50);

if (digitalRead(BTN\_TIME2\_DEC) == LOW && time2Value > 0) {

time2Value--;

showNumber(srTime2, time2Value);

while (digitalRead(BTN\_TIME2\_DEC) == LOW);

}

}

// Controle do cronômetro

if (digitalRead(BTN\_START\_STOP) == LOW) {

delay(200);

running = !running;

while (digitalRead(BTN\_START\_STOP) == LOW);

}

if (digitalRead(BTN\_ADD\_TIME) == LOW) {

delay(200);

if (maxTime < 180) {

maxTime += 60;

if (!running && seconds >= maxTime - 60) {

running = true;

}

}

while (digitalRead(BTN\_ADD\_TIME) == LOW);

}

if (digitalRead(BTN\_RESET) == LOW) {

delay(200);

running = false;

seconds = 0;

maxTime = defaultMaxTime;

while (digitalRead(BTN\_RESET) == LOW);

}

// Atualização do cronômetro

unsigned long currentMillis = millis();

if (running && currentMillis - previousMillis >= 1000) {

previousMillis = currentMillis;

if (seconds < maxTime) {

seconds++;

} else {

running = false;

}

}

displayTime(seconds);

}

# **6.5 Código via Bluetooth.**

#include <ShiftRegister74HC595.h>

#include <TM1637Display.h>

#include <SoftwareSerial.h>

// ==== Comunicação Bluetooth ====

SoftwareSerial bluetooth(6, 7); // RX, TX (ligue o TX do HC-06 no pino 6 e RX no 7)

// ==== Pinos do 74HC595 ====

#define SDI\_ROUND 39

#define SCLK\_ROUND 41

#define LOAD\_ROUND 43

#define SDI\_TIME1 48

#define SCLK\_TIME1 50

#define LOAD\_TIME1 52

#define SDI\_TIME2 8

#define SCLK\_TIME2 9

#define LOAD\_TIME2 10

// ==== Pinos do TM1637 ====

#define CLK 2

#define DIO 3

// ==== Botões Round ====

#define BTN\_ROUND\_INC 22

#define BTN\_ROUND\_DEC 23

// ==== Botões Time 1 ====

#define BTN\_TIME1\_INC 26

#define BTN\_TIME1\_DEC 27

// ==== Botões Time 2 ====

#define BTN\_TIME2\_INC 30

#define BTN\_TIME2\_DEC 31

// ==== Botões do Cronômetro ====

#define BTN\_START\_STOP 35

#define BTN\_ADD\_TIME 34

#define BTN\_RESET 44

// ==== Instâncias ====

ShiftRegister74HC595<2> srRound(SDI\_ROUND, SCLK\_ROUND, LOAD\_ROUND);

ShiftRegister74HC595<2> srTime1(SDI\_TIME1, SCLK\_TIME1, LOAD\_TIME1);

ShiftRegister74HC595<2> srTime2(SDI\_TIME2, SCLK\_TIME2, LOAD\_TIME2);

TM1637Display display(CLK, DIO);

// ==== Dígitos 7 segmentos ====

uint8\_t digits[] = {

B11000000, B11111001, B10100100, B10110000, B10011001,

B10010010, B10000010, B11111000, B10000000, B10010000

};

// ==== Variáveis ====

int roundValue = 0;

int time1Value = 0;

int time2Value = 0;

unsigned long previousMillis = 0;

bool running = false;

int defaultMaxTime = 120;

int maxTime = 120;

int seconds = 0;

// ==== Funções ====

void showNumber(ShiftRegister74HC595<2>& sr, int num) {

int unidades = num % 10;

int dezenas = (num / 10) % 10;

uint8\_t toPrint[] = {digits[dezenas], digits[unidades]};

sr.setAll(toPrint);

}

void displayTime(int sec) {

int minutes = sec / 60;

int secRemain = sec % 60;

int timeToDisplay = (minutes \* 100) + secRemain;

display.showNumberDecEx(timeToDisplay, 0x40, true);

}

// ==== Setup ====

void setup() {

bluetooth.begin(9600);

pinMode(BTN\_ROUND\_INC, INPUT\_PULLUP);

pinMode(BTN\_ROUND\_DEC, INPUT\_PULLUP);

pinMode(BTN\_TIME1\_INC, INPUT\_PULLUP);

pinMode(BTN\_TIME1\_DEC, INPUT\_PULLUP);

pinMode(BTN\_TIME2\_INC, INPUT\_PULLUP);

pinMode(BTN\_TIME2\_DEC, INPUT\_PULLUP);

pinMode(BTN\_START\_STOP, INPUT\_PULLUP);

pinMode(BTN\_ADD\_TIME, INPUT\_PULLUP);

pinMode(BTN\_RESET, INPUT\_PULLUP);

display.setBrightness(7);

display.showNumberDecEx(0, 0x40, true);

showNumber(srRound, roundValue);

showNumber(srTime1, time1Value);

showNumber(srTime2, time2Value);

}

// ==== Loop ====

void loop() {

// --- Controle por botões físicos ---

// Round

if (digitalRead(BTN\_ROUND\_INC) == LOW) {

delay(50);

if (digitalRead(BTN\_ROUND\_INC) == LOW && roundValue < 99) {

roundValue++;

showNumber(srRound, roundValue);

while (digitalRead(BTN\_ROUND\_INC) == LOW);

}

}

if (digitalRead(BTN\_ROUND\_DEC) == LOW) {

delay(50);

if (digitalRead(BTN\_ROUND\_DEC) == LOW && roundValue > 0) {

roundValue--;

showNumber(srRound, roundValue);

while (digitalRead(BTN\_ROUND\_DEC) == LOW);

}

}

// Time 1

if (digitalRead(BTN\_TIME1\_INC) == LOW) {

delay(50);

if (digitalRead(BTN\_TIME1\_INC) == LOW && time1Value < 99) {

time1Value++;

showNumber(srTime1, time1Value);

while (digitalRead(BTN\_TIME1\_INC) == LOW);

}

}

if (digitalRead(BTN\_TIME1\_DEC) == LOW) {

delay(50);

if (digitalRead(BTN\_TIME1\_DEC) == LOW && time1Value > 0) {

time1Value--;

showNumber(srTime1, time1Value);

while (digitalRead(BTN\_TIME1\_DEC) == LOW);

}

}

// Time 2

if (digitalRead(BTN\_TIME2\_INC) == LOW) {

delay(50);

if (digitalRead(BTN\_TIME2\_INC) == LOW && time2Value < 99) {

time2Value++;

showNumber(srTime2, time2Value);

while (digitalRead(BTN\_TIME2\_INC) == LOW);

}

}

if (digitalRead(BTN\_TIME2\_DEC) == LOW) {

delay(50);

if (digitalRead(BTN\_TIME2\_DEC) == LOW && time2Value > 0) {

time2Value--;

showNumber(srTime2, time2Value);

while (digitalRead(BTN\_TIME2\_DEC) == LOW);

}

}

// Cronômetro

if (digitalRead(BTN\_START\_STOP) == LOW) {

delay(200);

running = !running;

while (digitalRead(BTN\_START\_STOP) == LOW);

}

if (digitalRead(BTN\_ADD\_TIME) == LOW) {

delay(200);

if (maxTime < 180) {

maxTime += 60;

if (!running && seconds >= maxTime - 60) {

running = true;

}

}

while (digitalRead(BTN\_ADD\_TIME) == LOW);

}

if (digitalRead(BTN\_RESET) == LOW) {

delay(200);

running = false;

seconds = 0;

maxTime = defaultMaxTime;

while (digitalRead(BTN\_RESET) == LOW);

}

// --- Atualização do cronômetro ---

unsigned long currentMillis = millis();

if (running && currentMillis - previousMillis >= 1000) {

previousMillis = currentMillis;

if (seconds < maxTime) {

seconds++;

} else {

running = false;

}

}

displayTime(seconds);

// --- Controle via Bluetooth ---

if (bluetooth.available()) {

char comando = bluetooth.read();

switch (comando) {

case 'A': if (roundValue < 99) roundValue++; break;

case 'B': if (roundValue > 0) roundValue--; break;

case 'C': if (time1Value < 99) time1Value++; break;

case 'D': if (time1Value > 0) time1Value--; break;

case 'E': if (time2Value < 99) time2Value++; break;

case 'F': if (time2Value > 0) time2Value--; break;

case 'G': running = !running; break;

case 'H':

if (maxTime < 180) {

maxTime += 60;

if (!running && seconds >= maxTime - 60) running = true;

}

break;

case 'I':

running = false;

seconds = 0;

maxTime = defaultMaxTime;

break;

}

// Atualiza displays

showNumber(srRound, roundValue);

showNumber(srTime1, time1Value);

showNumber(srTime2, time2Value);

}

}