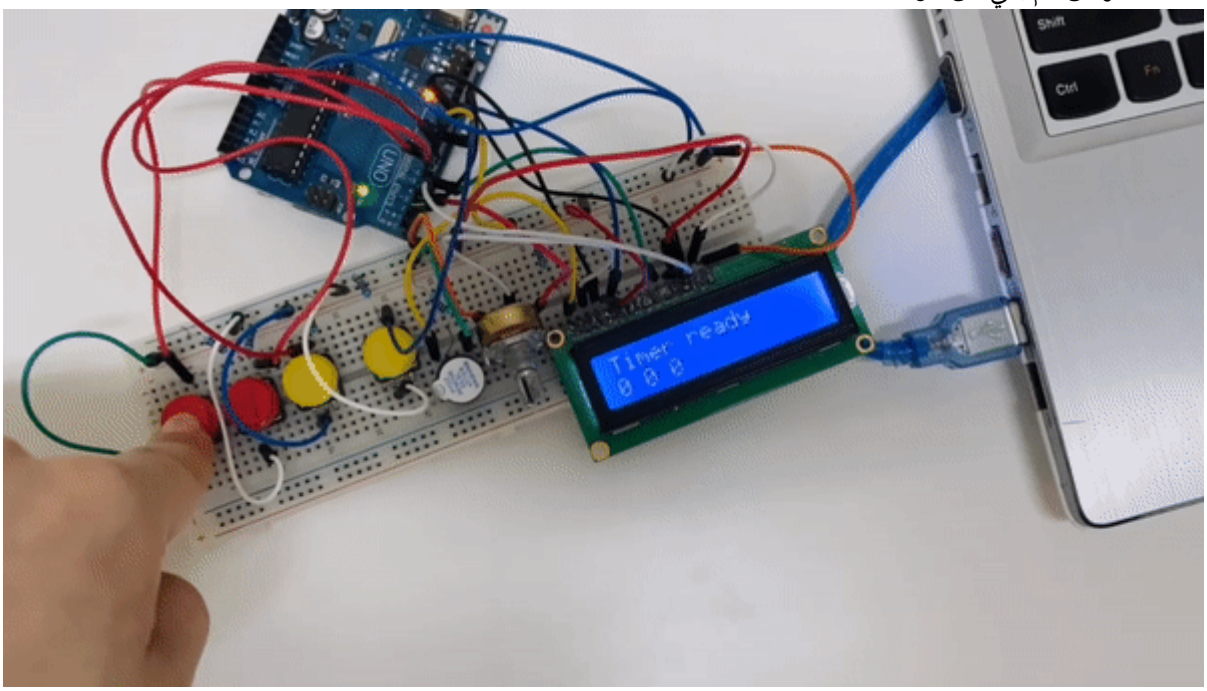
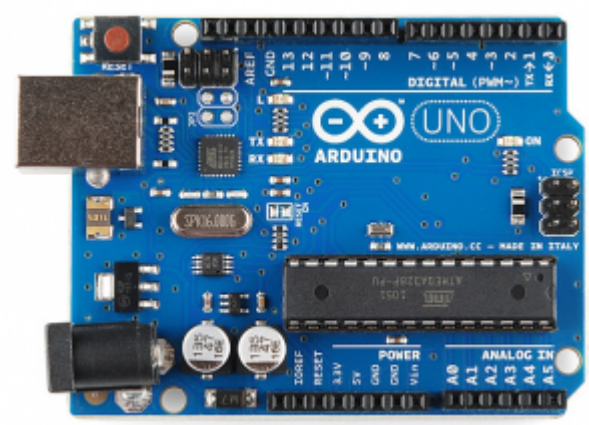
مؤقت باستخدام الاردوينو

مقدمة

في هذا الدرس ستتعلم كيف تصنع منبه باستخدام الاردوينو ومفاتيح تحكم تمكنك من ضبط الوقت، بالساعات والدقائق والثواني، مع شاشة لعرض كم بقي من الوقت .

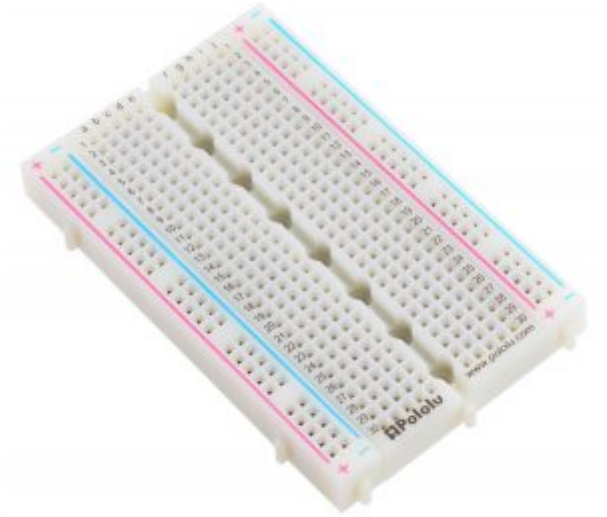


المواد والأدوات



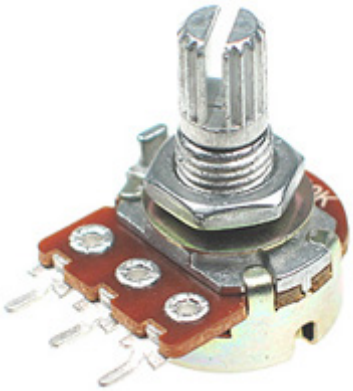
1× اردوينو اونو

1× سلك الاردوينو



1× لوحة تجارب – حجم كبير

1× شاشة كرستالية (LCD 2×16)



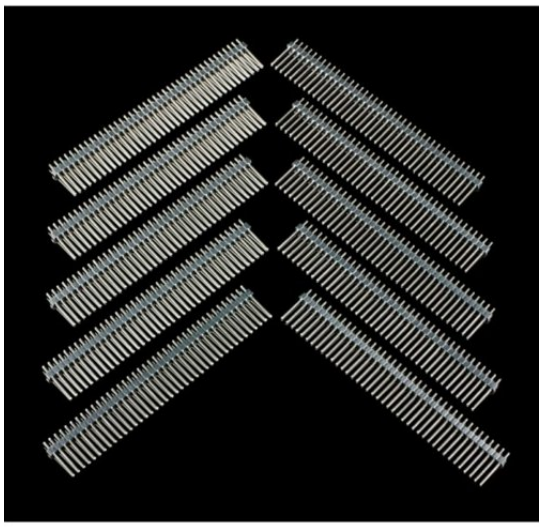
1× مقاومة متغيرة

حزمة أسلاك توصيل (ذكر- ذكر)

5× مقاومة 220 Ω

4× أزرار

1× مصدر صوت

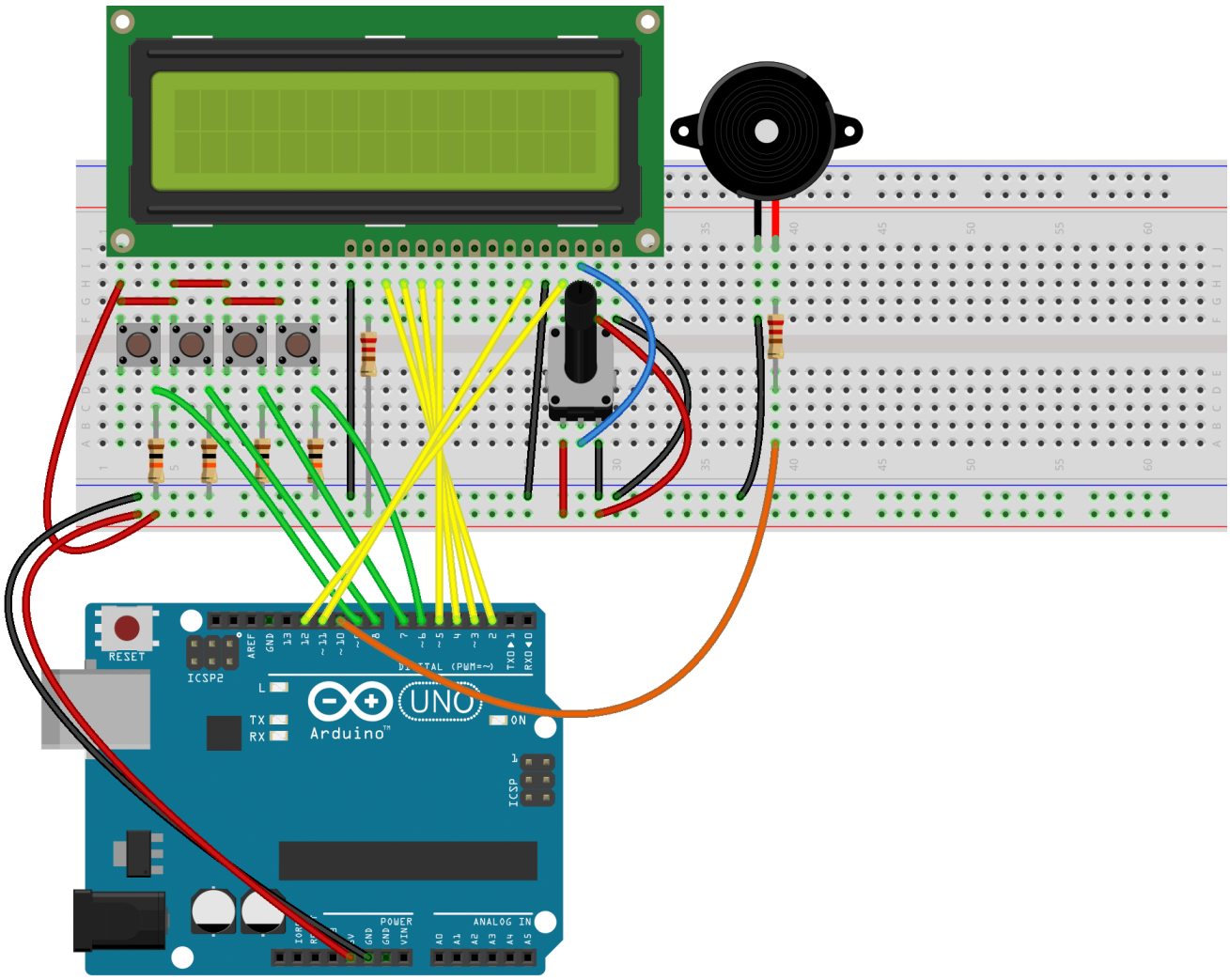


1× 40 رأس دبوس

توصيل الدائرة

لمعرفة المزيد حول الشاشة الكرستالية يمكنك الرجوع للدرس التحكم بالشاشة الكرستالية LCD

لابد من تلحيم المنافذ مع الشاشة الكرستالية، للمزيد حول اللحام يمكنك الرجوع للدرس تعلم كيفية التلحيم – تلحيم القطع باللوحة الإلكترونية



الكود البرمجي

في البداية عليك تحميل مكتبة الوقت.

ارفع الكود البرمجي على لوحة الاردوينو باستخدام برنامج اردوينو (IDE).

```
#include <LiquidCrystal.h>
#include <TimeLib.h> // FIX 2018-08-12 This fixes «‘now’ was not declared in this
scope» error

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

const int buzzerPin = 10;
const int resetButtonPin = 6;
const int startStopButtonPin = 7;
```

```
const int downButtonPin = 8;
const int upButtonPin = 9;

int setupHours = 0; // How many hours will count down when started
int setupMinutes = 0; // How many minutes will count down when started
int setupSeconds = 0; // How many seconds will count down when started
time_t setupTime = 0;


int currentHours = 0;
int currentMinutes = 0;
int currentSeconds = 0;
time_t currentTime = 0;

time_t startTime = 0;
time_t elapsedTime = 0;


int resetButtonState = LOW;
long resetButtonLongPressCounter = 0;
int startStopButtonState = LOW;
int upButtonState = LOW;
int downButtonState = LOW;
int resetButtonPrevState = LOW;
int startStopButtonPrevState = LOW;
int upButtonPrevState = LOW;
int downButtonPrevState = LOW;
bool resetButtonPressed = false;
bool resetButtonLongPressed = false;
bool startStopButtonPressed = false;
bool upButtonPressed = false;
bool downButtonPressed = false;


const int MODE_IDLE = 0;
const int MODE_SETUP = 1;
const int MODE_RUNNING = 2;
const int MODE_RINGING = 3;


int currentMode = MODE_IDLE; // 0=idle 1=setup 2=running 3=ringing
// Power up --> idle
// Reset --> idle
// Start/Stop --> start or stop counter
// Up / Down --> NOP
// Reset (long press) --> enter setup
// Start/Stop --> data select
// Up --> increase current data value
// Down --> decrease current data value
// Reset --> exit setup (idle)

int dataSelection = 0; // Currently selected data for edit (setup mode, changes with
Start/Stop)
```

```
// 0=hours (00-99) 1=minutes (00-59) 2=seconds (00-59)


void setup()
{
// put your setup code here, to run once:
lcd.begin(16, 2);
pinMode(resetButtonPin, INPUT);
pinMode(startStopButtonPin, INPUT);
pinMode(upButtonPin, INPUT);
pinMode(downButtonPin, INPUT);
pinMode(buzzerPin, OUTPUT);
Serial.begin(9600);
}

void loop()
{
// put your main code here, to run repeatedly:
startStopButtonPressed = false;
upButtonPressed = false;
downButtonPressed = false;
resetButtonPressed = false;
resetButtonLongPressed = false;
resetButtonState = digitalRead(resetButtonPin);
if(resetButtonState != resetButtonPrevState)
{
resetButtonPressed = resetButtonState == HIGH;
resetButtonPrevState = resetButtonState;
}
else // Long press management...
{
if(resetButtonState == HIGH)
{
resetButtonLongPressCounter++;
if(resetButtonLongPressCounter == 100)
{
resetButtonPressed = false;
resetButtonLongPressed = true;
resetButtonLongPressCounter = 0;
}
}
else
{
resetButtonLongPressCounter = 0;
resetButtonPressed = false;
resetButtonLongPressed = false;
}
}
startStopButtonPressed = false;
startStopButtonState = digitalRead(startStopButtonPin);
if(startStopButtonState != startStopButtonPrevState)
{
startStopButtonPressed = startStopButtonState == HIGH;
```

```
startStopButtonPrevState = startStopButtonState;
}
downButtonPressed = false;
downButtonState = digitalRead(downButtonPin);
if(downButtonState != downButtonPrevState)
{

downButtonPressed = downButtonState == HIGH;
downButtonPrevState = downButtonState;
}
upButtonPressed = false;
upButtonState = digitalRead(upButtonPin);
if(upButtonState != upButtonPrevState)
{
upButtonPressed = upButtonState == HIGH;
upButtonPrevState = upButtonState;
}
switch(currentMode)
{
case MODE_IDLE:
if(resetButtonPressed)
{
Reset();
}
if(resetButtonLongPressed)
{
currentMode = MODE_SETUP;
}
if(startStopButtonPressed)
{
currentMode = currentMode == MODE_IDLE ? MODE_RUNNING : MODE_IDLE;
if(currentMode == MODE_RUNNING)
{
startTime = now();
}
}
break;
case MODE_SETUP:
if(resetButtonPressed)
{
// Exit setup mode
setupTime = setupSeconds + (60 * setupMinutes) + (3600 * setupHours);
currentHours = setupHours;
currentMinutes = setupMinutes;
currentSeconds = setupSeconds;
dataSelection = 0;
currentMode = MODE_IDLE;
}
if(startStopButtonPressed)
{
dataSelection++;
if(dataSelection == 3)
{
```

```
dataSelection = 0;
}
}
if(downButtonPressed)
{
switch(dataSelection)
{
case 0: // hours
setupHours--;
if(setupHours == -1)
{
setupHours = 99;
}
break;
case 1: // minutes
setupMinutes--;
if(setupMinutes == -1)
{
setupMinutes = 59;
}
break;
case 2: // seconds
setupSeconds--;
if(setupSeconds == -1)
{
setupSeconds = 59;
}
break;
}
}
if(upButtonPressed)
{
switch(dataSelection)
{
case 0: // hours
setupHours++;
if(setupHours == 100)
{
setupHours = 0;
}
break;
case 1: // minutes
setupMinutes++;
if(setupMinutes == 60)
{
setupMinutes = 0;
}
break;
case 2: // seconds
setupSeconds++;
if(setupSeconds == 60)
{
setupSeconds = 0;
```

```
}
break;
}
}
break;
case MODE_RUNNING:
if(startStopButtonPressed)
{
currentMode = MODE_IDLE;
}
if(resetButtonPressed)
{
Reset();
currentMode = MODE_IDLE;
}
break;
case MODE_RINGING:
if(resetButtonPressed || startStopButtonPressed || downButtonPressed ||
upButtonPressed)
{
currentMode = MODE_IDLE;
}
break;
}
switch(currentMode)
{
case MODE_IDLE:
case MODE_SETUP:
// NOP
break;
case MODE_RUNNING:
currentTime = setupTime - (now() - startTime);
if(currentTime <= 0)
{
currentMode = MODE_RINGING;
}
break;
case MODE_RINGING:
analogWrite(buzzerPin, 20);
delay(20);
analogWrite(buzzerPin, 0);
delay(40);
break;
}
//lcd.clear();
lcd.setCursor(0, 0);
switch(currentMode)
{
case MODE_IDLE:
lcd.print("Timer ready ");
lcd.setCursor(0, 1);
lcd.print(currentHours);
lcd.print(" ");
```

```
lcd.print(currentMinutes);
lcd.print(" ");
lcd.print(currentSeconds);
lcd.print(" ");
break;
case MODE_SETUP:
lcd.print("Setup mode: ");
switch(dataSelection)
{
case 0:
lcd.print("HRS ");
break;
case 1:
lcd.print("MINS");
break;
case 2:
lcd.print("SECS");
break;
}
lcd.setCursor(0, 1);
lcd.print(setupHours);
lcd.print(" ");
lcd.print(setupMinutes);
lcd.print(" ");
lcd.print(setupSeconds);
lcd.print(" ");
break;
case MODE_RUNNING:
lcd.print("Counting down...");
lcd.setCursor(0, 1);
if(hour(currentTime) < 10) lcd.print("0");
lcd.print(hour(currentTime));
lcd.print(":");
if(minute(currentTime) < 10) lcd.print("0");
lcd.print(minute(currentTime));
lcd.print(":");
if(second(currentTime) < 10) lcd.print("0");
lcd.print(second(currentTime));


break;
case MODE_RINGING:
lcd.print(" RING-RING! ");
lcd.setCursor(0, 1);
lcd.print(" ");
break;
}
delay(10);
}

void Reset()
{
currentMode = MODE_IDLE;
```

```
currentHours = setupHours;
currentMinutes = setupMinutes;
currentSeconds = setupSeconds;
}
```

شرح الكود البرمجي

هذا السطر يستدعي مكتبة الشاشة الكرستالية.

نستطيع تحميلها بتتبع المسار التالي:

Sketch > Include libraries > Manage libraries

ثم نكتب بخانة البحث Liquid crystal by Arduino

ثم نضغط على Install.

```
#include <LiquidCrystal.h>
```

هنا نستدعي مكتبة الوقت.

```
#include <TimeLib.h>
```

بعد ذلك عرفنا المتغيرات الخاصة بالشاشة الكرستالية.

```
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
```

هذه الأسطر توضح منافذ الاردوينو التي ستستخدمها للربط في هذا المشروع.

المنفذ الرقمي 10 يتم ربطه مع مصدر الصوت.

المنفذ الرقمي 6 يتم ربطه مع زر الاختيار.

المنفذ الرقمي 7 يتم ربطه مع زر التشغيل والإيقاف.

المنفذ الرقمي 8 يتم ربطه مع زر العد التنازلي.

المنفذ الرقمي 9 يتم ربطه مع زر العد التصاعدي.

```
const int buzzerPin = 10;
const int resetButtonPin = 6;
const int startStopButtonPin = 7;
const int downButtonPin = 8;
const int upButtonPin = 9;
```

في هذه الأسطر قمنا بإنشاء متغيرات لتخزين الحالة الابتدائية للأزرار وتساوي صفر.

```
int setupHours = 0; // How many hours will count down when started
int setupMinutes = 0; // How many minutes will count down when started
```

```
int setupSeconds = 0; // How many seconds will count down when started
time_t setupTime = 0;
```

في هذه الأسطر قمنا بإنشاء متغيرات لتخزين الحالة الحالية للأزرار وتساوي صفر.

```
int currentHours = 0;
int currentMinutes = 0;
int currentSeconds = 0;
time_t currentTime = 0;
```

هنا سيتم تعريف أكثر من حالة للأزرار سيتم الخلط بينها بناء على المدخلات على لوحة الاردوينو.

```
int resetButtonState = LOW;
long resetButtonLongPressCounter = 0;
int startStopButtonState = LOW;
int upButtonState = LOW;
int downButtonState = LOW;
int resetButtonPrevState = LOW;
int startStopButtonPrevState = LOW;
int upButtonPrevState = LOW;
int downButtonPrevState = LOW;
bool resetButtonPressed = false;
bool resetButtonLongPressed = false;
bool startStopButtonPressed = false;
bool upButtonPressed = false;
bool downButtonPressed = false;
```

هنا يتم تعريف الأوضاع الأربع للنظام.

```
const int MODE_IDLE = 0;
const int MODE_SETUP = 1;
const int MODE_RUNNING = 2;
const int MODE_RINGING = 3;
```

يتم تخزين الحالة الابتدائية للنظام تساوي صفر؛ لأن المستخدم لم يقم بالاختيار بعد.

```
int dataSelection = 0; // Currently selected data for edit (setup mode, changes with
Start/Stop)
```

في الدالة ()setup يتم تهيئة الشاشة الكرستالية وتهيئة الأزرار كمدخلات ومصدر الصوت والشاشة كمخرجات.

```
void setup()
{
// put your setup code here, to run once:
lcd.begin(16, 2);
pinMode(resetButtonPin, INPUT);
pinMode(startStopButtonPin, INPUT);
pinMode(upButtonPin, INPUT);
pinMode(downButtonPin, INPUT);
pinMode(buzzerPin, OUTPUT);
```

```
Serial.begin(9600);
}
```

في الدالة ()loop سيتم برمجة الأزرار.

في حال ضغط المستخدم بشكل مستمر على زر الاختيار سيتم عرض على الشاشة الكرستالية 3 خيارات يمكن للمستخدم ضبطهما: الساعات والدقائق والثواني.

يمكن للمستخدم ضبط الساعات أو الدقائق أو الثواني أو جميعها بنفس الوقت من خلال الضغط على زر التشغيل والإيقاف.

يمكن التحكم بالوقت تنازليًا من خلال زر العد التنازلي.

ويمكن التحكم بالوقت تصاعديًا من خلال زر العد التصاعدي.

بعد اختيار الوقت المناسب اضغط زر الاختيار لتأكيد ذلك.

ثم اضغط على زر التشغيل والإيقاف للبدء.

بعد انتهاء الوقت المحسوب سيبدأ مصدر الصوت بالعمل يمكنك إيقافه من خلال زر الإيقاف والتشغيل.

```
void loop()
{
// put your main code here, to run repeatedly:
startStopButtonPressed = false;
upButtonPressed = false;
downButtonPressed = false;
resetButtonPressed = false;
resetButtonLongPressed = false;
resetButtonState = digitalRead(resetButtonPin);
if(resetButtonState != resetButtonPrevState)
{
resetButtonPressed = resetButtonState == HIGH;
resetButtonPrevState = resetButtonState;
}
else // Long press management...
{
if(resetButtonState == HIGH)
{
resetButtonLongPressCounter++;
if(resetButtonLongPressCounter == 100)
{
resetButtonPressed = false;
resetButtonLongPressed = true;
resetButtonLongPressCounter = 0;
}
}
else
{
resetButtonLongPressCounter = 0;
resetButtonPressed = false;
resetButtonLongPressed = false;
}
}
```

```
startStopButtonPressed = false;
startStopButtonState = digitalRead(startStopButtonPin);
if(startStopButtonState != startStopButtonPrevState)
{
startStopButtonPressed = startStopButtonState == HIGH;
startStopButtonPrevState = startStopButtonState;
}
downButtonPressed = false;
downButtonState = digitalRead(downButtonPin);
if(downButtonState != downButtonPrevState)
{

downButtonPressed = downButtonState == HIGH;
downButtonPrevState = downButtonState;
}
upButtonPressed = false;
upButtonState = digitalRead(upButtonPin);
if(upButtonState != upButtonPrevState)
{
upButtonPressed = upButtonState == HIGH;
upButtonPrevState = upButtonState;
}
switch(currentMode)
{
case MODE_IDLE:
if(resetButtonPressed)
{
Reset();
}
if(resetButtonLongPressed)
{
currentMode = MODE_SETUP;
}
if(startStopButtonPressed)
{
currentMode = currentMode == MODE_IDLE ? MODE_RUNNING : MODE_IDLE;
if(currentMode == MODE_RUNNING)
{
startTime = now();
}
}
break;
case MODE_SETUP:
if(resetButtonPressed)
{
// Exit setup mode
setupTime = setupSeconds + (60 * setupMinutes) + (3600 * setupHours);
currentHours = setupHours;
currentMinutes = setupMinutes;
currentSeconds = setupSeconds;
dataSelection = 0;
currentMode = MODE_IDLE;
}
```

```
if(startStopButtonPressed)
{
dataSelection++;
if(dataSelection == 3)
{
dataSelection = 0;
}
}
if(downButtonPressed)
{
switch(dataSelection)
{
case 0: // hours
setupHours--;
if(setupHours == -1)
{
setupHours = 99;
}
break;
case 1: // minutes
setupMinutes--;
if(setupMinutes == -1)
{
setupMinutes = 59;
}
break;
case 2: // seconds
setupSeconds--;
if(setupSeconds == -1)
{
setupSeconds = 59;
}
break;
}
}
if(upButtonPressed)
{
switch(dataSelection)
{
case 0: // hours
setupHours++;
if(setupHours == 100)
{
setupHours = 0;
}
break;
case 1: // minutes
setupMinutes++;
if(setupMinutes == 60)
{
setupMinutes = 0;
}
break;
```

```
case 2: // seconds
setupSeconds++;
if(setupSeconds == 60)
{
setupSeconds = 0;
}
break;
}
}
break;
case MODE_RUNNING:
if(startStopButtonPressed)
{
currentMode = MODE_IDLE;
}
if(resetButtonPressed)
{
Reset();
currentMode = MODE_IDLE;
}
break;
case MODE_RINGING:
if(resetButtonPressed || startStopButtonPressed || downButtonPressed ||
upButtonPressed)
{
currentMode = MODE_IDLE;
}
break;
}
switch(currentMode)
{
case MODE_IDLE:
case MODE_SETUP:
// NOP
break;
case MODE_RUNNING:
currentTime = setupTime - (now() - startTime);
if(currentTime <= 0)
{
currentMode = MODE_RINGING;
}
break;
case MODE_RINGING:
analogWrite(buzzerPin, 20);
delay(20);
analogWrite(buzzerPin, 0);
delay(40);
break;
}
//lcd.clear();
lcd.setCursor(0, 0);
switch(currentMode)
{
```

```
case MODE_IDLE:
lcd.print("Timer ready ");
lcd.setCursor(0, 1);
lcd.print(currentHours);
lcd.print(" ");
lcd.print(currentMinutes);
lcd.print(" ");
lcd.print(currentSeconds);
lcd.print(" ");
break;
case MODE_SETUP:
lcd.print("Setup mode: ");
switch(dataSelection)
{
case 0:
lcd.print("HRS ");
break;
case 1:
lcd.print("MINS");
break;
case 2:
lcd.print("SECS");
break;
}
lcd.setCursor(0, 1);
lcd.print(setupHours);
lcd.print(" ");
lcd.print(setupMinutes);
lcd.print(" ");
lcd.print(setupSeconds);
lcd.print(" ");
break;
case MODE_RUNNING:
lcd.print("Counting down...");
lcd.setCursor(0, 1);
if(hour(currentTime) < 10) lcd.print("0");
lcd.print(hour(currentTime));
lcd.print(":");
if(minute(currentTime) < 10) lcd.print("0");
lcd.print(minute(currentTime));
lcd.print(":");
if(second(currentTime) < 10) lcd.print("0");
lcd.print(second(currentTime));


break;
case MODE_RINGING:
lcd.print(" RING-RING! ");
lcd.setCursor(0, 1);
lcd.print(" ");
break;
}
delay(10);
```

```
}

void Reset()
{
currentMode = MODE_IDLE;
currentHours = setupHours;
currentMinutes = setupMinutes;
currentSeconds = setupSeconds;
}
```

يمكنك اختبار مؤقت العد بعد رفع الكود البرمجي.

لا تنسَ فصل مصدر الطاقة بعد الانتهاء من استخدام النظام.