Практическое занятие №
Тема: «4 разрядный 7-сегментный индикатор и сдвиговый регистр 74HC595N»

Цель работы: приобрести практические навыки по подключению и программированию 7-сегментного индикатора и сдвигового регистра 74HC595N на платформе Arduino.

Последовательность выполнения работы:

– Собрать схемы на макетной плате, иначе при отсутствии набора Arduino в web-приложениях (https://wokwi.com/projects/new/arduino-uno или https://www.tinkercad.com/) для приведенных примеров.

– Запрограммировать микроконтроллер согласно заданию в примере.

– Выполнить задание для самостоятельной работы.

Содержание отчета:

– Название практического занятия, его цель.

– Фото или скриншоты собранной схемы.

– Написанный программный код вставить текстом, Courier New, 12 кегль, одинарный отступ без абзацев.

– Вывод о проделанной работе.

– Файл Fritzing с принципиальной и монтажной схемой.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

4-разрядный 7-сегментный индикатор

4-разрядный 7-сегментный индикатор — это электронное устройство, представляющее собой дисплей с четырьмя отдельными 7-сегментными индикаторами. Каждый индикатор может отображать цифры от 0 до 9, а также некоторые буквы.

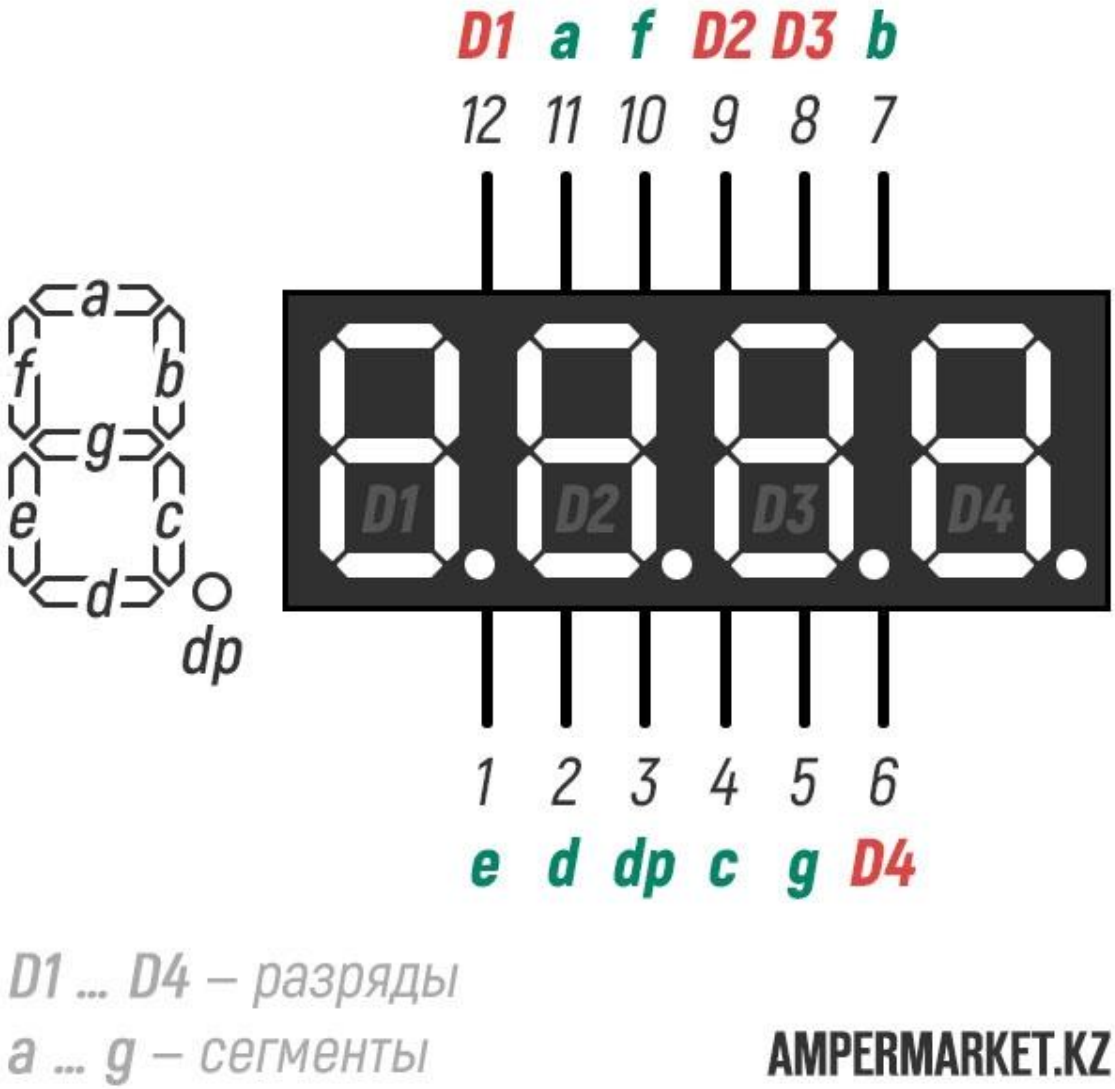


Рисунок 1 – 4 разрядный 7-сегментный цифровой LED индикатор

Восьмиразрядный сдвиговый регистр

74HC595N – восьмиразрядный сдвиговый регистр с последовательным вводом, последовательным или параллельным выводом информации, с триггером-защелкой и тремя состояниями на выходе. Самое распространенное применение данного регистра – экономия выходов микроконтроллера. Данный сдвиговый регистр позволяет управлять напряжением на своих восьми выходах, заняв всего три выхода микроконтроллера. Таким образом количество рабочих выводов увеличивается на пять.

Кроме того, регистры 74HC595 можно подключать каскадом один за другим (через пин **Q7'**), и таким образом из всё тех же 3 входящих линий получать 16, 24, 32 и т.д. цифровых выходов.

74HC595N имеет следующие входы:

– **[10] MR** — сброс регистра, при подаче логического нуля на MR и единицы на STCP переводит все выходы в состояние логического нуля;

– **[11] SH_CP** — вход для тактовых импульсов;

– **[12] ST_CP** — линия прерываний;

– **[13] OE** — вход, переводящий выходы из высокоимпедансного состояния в рабочее;

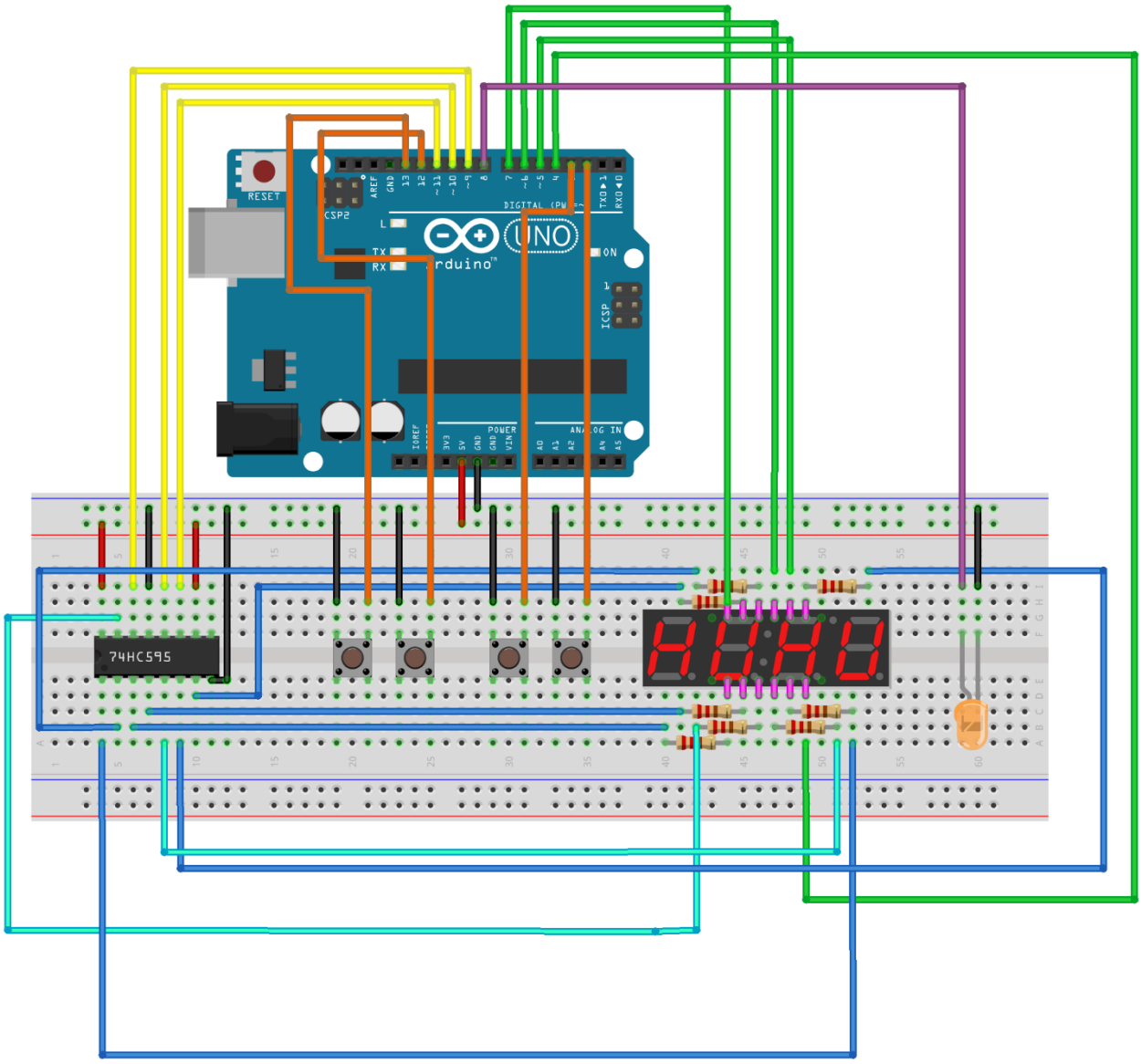
– **[14] DS** — вход данных;

– **[8] GND** — Ground. Земля

– **[16] VCC** — Питание +5 В.

Рисунок 2 – Восьмиразрядный сдвиговый регистр

ЗАДАНИЯ



Задание 1: Таймер на 10 минут с управлением кнопками

```
#include <Arduino.h>
// Пины для управления сдвиговым регистром
const int dataPin = 9;
const int clockPin = 11;
const int latchPin = 10;

// Пины для разрядов индикатора
const int digitPins[] = {7, 6, 5, 4};

// Пины кнопок
const int btnStop = 2;
const int btnStart = 3;
```

```
// Массив для отображения цифр (0-9) на семисегментном индикаторе
byte digitPatterns[10] = {
  B11111100, // 0
  B01100000, // 1
  B11011010, // 2
  B11110010, // 3
  B01100110, // 4
  B10110110, // 5
  B10111110, // 6
  B11100000, // 7
  B11111110, // 8
  B11110110  // 9
};

volatile unsigned long totalSeconds = 600; // 10 минут
volatile boolean isRunning = true;
unsigned long lastTime = 0;

void setup() {
  // Настройка пинов
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(latchPin, OUTPUT);

  for (int i = 0; i < 4; i++) {
    pinMode(digitPins[i], OUTPUT);
    digitalWrite(digitPins[i], HIGH);
  }

  // Настройка кнопок с подтяжкой
  pinMode(btnStop, INPUT_PULLUP);
  pinMode(btnStart, INPUT_PULLUP);

   // Прерывания для кнопок
   attachInterrupt(digitalPinToInterrupt(btnStop), stopTimer,
FALLING);
  attachInterrupt(digitalPinToInterrupt(btnStart),
startTimer, FALLING);
}

void loop() {
  if (isRunning && millis() - lastTime >= 1000) {
    lastTime = millis();
    if (totalSeconds > 0) {
      totalSeconds--;
    } else {
      totalSeconds = 600; // Автоматический перезапуск
    }
  }
  displayTime();
}
```

```
void displayTime() {
  int minutes = totalSeconds / 60;
  int seconds = totalSeconds % 60;

  int digits[4] = {
    minutes / 10,
    minutes % 10,
    seconds / 10,
    seconds % 10
  };

  for (int i = 0; i < 4; i++) {
    digitalWrite(latchPin, LOW);
    shiftOut(dataPin,          clockPin,          LSBFIRST,
digitPatterns[digits[i]]);

    for (int j = 0; j < 4; j++) {
      digitalWrite(digitPins[j], HIGH);
    }
    digitalWrite(digitPins[i], LOW);

    digitalWrite(latchPin, HIGH);
    delay(5);
  }
}

void stopTimer() {
  static unsigned long lastInterrupt = 0;
  if (millis() - lastInterrupt > 200) {
    isRunning = false;
  }
  lastInterrupt = millis();
}

void startTimer() {
  static unsigned long lastInterrupt = 0;
  if (millis() - lastInterrupt > 200) {
    isRunning = true;
  }
  lastInterrupt = millis();
}
```

2. Игра на реакцию

```
#include <Arduino.h>
// Пины управления
const int dataPin = 9;
const int clockPin = 11;
const int latchPin = 10;
const int digitPins[] = {7, 6, 5, 4};
const int ledPin = 8;
const int btnStart = 3;
const int btnReact = 2;
```

```
const int btnReset = 12;

byte digitPatterns[11] = {
  B11111100, // 0
  B01100000, // 1
  B11011010, // 2
  B11110010, // 3
  B01100110, // 4
  B10110110, // 5
  B10111110, // 6
  B11100000, // 7
  B11111110, // 8
  B11110110, // 9
  B00000000  // Пусто
};

enum State { WAITING, COUNTDOWN, RUNNING, RESULT };
State currentState = WAITING;

unsigned long randomTime;
unsigned long startTime;
unsigned long reactionTime;
byte currentDigit = 10;

void setup() {
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(latchPin, OUTPUT);

  for (int i = 0; i < 4; i++) {
    pinMode(digitPins[i], OUTPUT);
    digitalWrite(digitPins[i], HIGH);
  }

  pinMode(ledPin, OUTPUT);
  pinMode(btnStart, INPUT_PULLUP);
  pinMode(btnReact, INPUT_PULLUP);
  pinMode(btnReset, INPUT_PULLUP);

  randomSeed(analogRead(0));
}

void loop() {
  switch (currentState) {
    case WAITING:
      if (digitalRead(btnStart) == LOW) {
        randomTime = random(3000, 15000);
        currentState = COUNTDOWN;
        startTime = millis();
      }
      displayDigit(10, 3);
      break;
```

```
    case COUNTDOWN:
      if (millis() - startTime >= randomTime) {
        digitalWrite(ledPin, HIGH);
        currentState = RUNNING;
        startTime = millis();
      }
      displayDigit(((randomTime - (millis() - startTime)) /
1000) % 10, 3);
      break;

    case RUNNING:
      reactionTime = millis() - startTime;
      displayTime(reactionTime);

      if (digitalRead(btnReact) == LOW) {
        currentState = RESULT;
      }
      break;

    case RESULT:
      displayTime(reactionTime);
      if (digitalRead(btnReset) == LOW) {
        digitalWrite(ledPin, LOW);
        currentState = WAITING;
      }
      break;
  }
}

void displayTime(unsigned long time) {
  int digits[4] = {
    (time / 1000) % 10,
    (time / 100) % 10,
    (time / 10) % 10,
    time % 10
  };

  for (int i = 0; i < 4; i++) {
    digitalWrite(latchPin, LOW);
    byte pattern = digitPatterns[digits[i]];
    if (i == 1) pattern |= B00000001; // Добавляем точку
    shiftOut(dataPin, clockPin, LSBFIRST, pattern);

    for (int j = 0; j < 4; j++) {
      digitalWrite(digitPins[j], HIGH);
    }
    digitalWrite(digitPins[i], LOW);

    digitalWrite(latchPin, HIGH);
    delay(5);
  }
}
```

```
void displayDigit(byte digit, byte position) {
  digitalWrite(latchPin, LOW);
  shiftOut(dataPin,           clockPin,           LSBFIRST,
digitPatterns[digit]);

  for (int i = 0; i < 4; i++) {
    digitalWrite(digitPins[i], HIGH);
  }
  digitalWrite(digitPins[position], LOW);

  digitalWrite(latchPin, HIGH);
}
```

Задание 3. Программируемый таймер с настройкой

```
#include <Arduino.h>

// Пины управления
const int dataPin = 9;
const int clockPin = 11;
const int latchPin = 10;
const int digitPins[] = {7, 6, 5, 4};
const int btnPause = 2;
const int btnSelect = 3;
const int btnDec = 12;
const int btnInc = 13;

byte digitPatterns[10] = {
  B11111100, // 0
  B01100000, // 1
  B11011010, // 2
  B11110010, // 3
  B01100110, // 4
  B10110110, // 5
  B10111110, // 6
  B11100000, // 7
  B11111110, // 8
  B11110110  // 9
};

volatile int minutes = 0;
volatile int seconds = 0;
volatile bool isRunning = false;
volatile int selectedDigit = -1;
volatile unsigned long lastBlink = 0;
volatile bool blinkState = false;
volatile unsigned long lastPress = 0;
volatile bool reverseSelect = false;

void setup() {
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
```

```
  pinMode(latchPin, OUTPUT);

  for (int i = 0; i < 4; i++) {
    pinMode(digitPins[i], OUTPUT);
    digitalWrite(digitPins[i], HIGH);
  }

  pinMode(btnPause, INPUT_PULLUP);
  pinMode(btnSelect, INPUT_PULLUP);
  pinMode(btnDec, INPUT_PULLUP);
  pinMode(btnInc, INPUT_PULLUP);

  attachInterrupt(digitalPinToInterrupt(btnPause),
togglePause, FALLING);
  attachInterrupt(digitalPinToInterrupt(btnSelect),
changeDigit, FALLING);
}

void loop() {
  static unsigned long lastSecond = 0;

  if (isRunning && millis() - lastSecond >= 1000) {
    lastSecond = millis();
    if (seconds > 0) seconds--;
    else if (minutes > 0) {
      minutes--;
      seconds = 59;
    }
  }

  checkButtons();
  displayTime();
}

void displayTime() {
  int digits[4] = {minutes / 10, minutes % 10, seconds / 10,
seconds % 10};

  for (int i = 0; i < 4; i++) {
    if (selectedDigit == i && millis() - lastBlink > 500) {
      blinkState = !blinkState;
      lastBlink = millis();
    }

    digitalWrite(latchPin, LOW);
    if (selectedDigit == i && blinkState) {
      shiftOut(dataPin, clockPin, LSBFIRST, B00000000);
    } else {
      shiftOut(dataPin,          clockPin,          LSBFIRST,
digitPatterns[digits[i]]);
    }

    for (int j = 0; j < 4; j++) {
```

```
      digitalWrite(digitPins[j], HIGH);
    }
    digitalWrite(digitPins[i], LOW);

    digitalWrite(latchPin, HIGH);
    delay(5);
  }
}

void checkButtons() {
  static unsigned long lastChange = 0;

  if (selectedDigit >= 0) {
    if (digitalRead(btnDec) == LOW) {
      if (millis() - lastChange > 500) {
        changeValue(false);
        lastChange = millis();
      }
    }

    if (digitalRead(btnInc) == LOW) {
      if (millis() - lastChange > 500) {
        changeValue(true);
        lastChange = millis();
      }
    }
  }
}

void changeValue(bool increase) {
  int* values[4] = {&minutes, &minutes, &seconds, &seconds};
  int limits[4] = {5, 9, 5, 9};

  int currentValue = *values[selectedDigit];

  if (increase) {
    currentValue     =     (currentValue     +     1)     %
(limits[selectedDigit] + 1);
  } else {
    currentValue     =     (currentValue     ==     0)     ?
limits[selectedDigit] : currentValue - 1;
  }

  *values[selectedDigit] = currentValue;
}

void togglePause() {
  static unsigned long lastInterrupt = 0;
  if (millis() - lastInterrupt > 200) {
    isRunning = !isRunning;
  }
  lastInterrupt = millis();
}
```

```
void changeDigit() {
  static unsigned long lastInterrupt = 0;
  if (millis() - lastInterrupt > 200) {
    if (millis() - lastPress > 3000) {
      reverseSelect = !reverseSelect;
    }

    if (reverseSelect) {
      selectedDigit = (selectedDigit - 1 + 4) % 4;
    } else {
      selectedDigit = (selectedDigit + 1) % 4;
    }

    if (selectedDigit == -1) selectedDigit = 3;
  }
  lastInterrupt = millis();
  lastPress = millis();
}
```

Задание 4. Вывод даты и времени

```
#include <Arduino.h>

// Пины для управления сдвиговым регистром
const int dataPin = 9;
const int clockPin = 11;
const int latchPin = 10;

// Пины для разрядов индикатора
const int digitPins[] = {7, 6, 5, 4};

// Массив для отображения цифр (0-9) на семисегментном индикаторе
byte digitPatterns[10] = {
  B11111100, // 0
  B01100000, // 1
  B11011010, // 2
  B11110010, // 3
  B01100110, // 4
  B10110110, // 5
  B10111110, // 6
  B11100000, // 7
  B11111110, // 8
  B11110110  // 9
};

// Переменные для хранения времени и даты
int hours = 0;
int minutes = 0;
int day = 1;
int month = 1;
```

```
// Флаг для переключения между временем и датой
bool showTime = true;
unsigned long lastChangeTime = 0;
unsigned long changeInterval = 0;

void setup() {
  // Настройка пинов
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(latchPin, OUTPUT);

  for (int i = 0; i < 4; i++) {
    pinMode(digitPins[i], OUTPUT);
    digitalWrite(digitPins[i], HIGH);
  }

  // Получаем время компиляции (примерное время загрузки)
  // __TIME__ имеет формат "HH:MM:SS"
  String compileTime = __TIME__;
  hours = compileTime.substring(0, 2).toInt();
  minutes = compileTime.substring(3, 5).toInt();

  // __DATE__ имеет формат "MMM DD YYYY"
  String compileDate = __DATE__;
  String monthStr = compileDate.substring(0, 3);

  // Преобразуем текстовое представление месяца в число
  if (monthStr == "Jan") month = 1;
  else if (monthStr == "Feb") month = 2;
  else if (monthStr == "Mar") month = 3;
  else if (monthStr == "Apr") month = 4;
  else if (monthStr == "May") month = 5;
  else if (monthStr == "Jun") month = 6;
  else if (monthStr == "Jul") month = 7;
  else if (monthStr == "Aug") month = 8;
  else if (monthStr == "Sep") month = 9;
  else if (monthStr == "Oct") month = 10;
  else if (monthStr == "Nov") month = 11;
  else if (monthStr == "Dec") month = 12;

  day = compileDate.substring(4, 6).toInt();

  // Генерируем первый случайный интервал
  randomSeed(analogRead(0));
  changeInterval = random(10000, 20000); // 10-20 секунд в
миллисекундах
  lastChangeTime = millis();
}

void loop() {
  // Проверяем, не пришло ли время сменить отображение
  if (millis() - lastChangeTime >= changeInterval) {
    showTime = !showTime;
```

```
    lastChangeTime = millis();
    changeInterval = random(10000, 20000); // Новый случайный
интервал
  }

  // Обновляем отображение
  if (showTime) {
    displayTime();
  } else {
    displayDate();
  }
}

void displayTime() {
  int digits[4] = {
    hours / 10,
    hours % 10,
    minutes / 10,
    minutes % 10
  };

  for (int i = 0; i < 4; i++) {
    digitalWrite(latchPin, LOW);
    shiftOut(dataPin,          clockPin,          LSBFIRST,
digitPatterns[digits[i]]);

    for (int j = 0; j < 4; j++) {
      digitalWrite(digitPins[j], HIGH);
    }
    digitalWrite(digitPins[i], LOW);

    digitalWrite(latchPin, HIGH);
    delay(5);
  }
}

void displayDate() {
  int digits[4] = {
    day / 10,
    day % 10,
    month / 10,
    month % 10
  };

  for (int i = 0; i < 4; i++) {
    digitalWrite(latchPin, LOW);
    shiftOut(dataPin,          clockPin,          LSBFIRST,
digitPatterns[digits[i]]);

    for (int j = 0; j < 4; j++) {
      digitalWrite(digitPins[j], HIGH);
    }
    digitalWrite(digitPins[i], LOW);
```

```
    digitalWrite(latchPin, HIGH);
    delay(5);
  }
}
```