УДК 004.032 **DOI 10.52928/2070-1624-2023-40-1-17-22**

СТРЕЛОЧНЫЕ ПРИБОРЫ НА ШАГОВЫХ ДВИГАТЕЛЯХ С МИКРОКОНТРОЛЛЕРНЫМ УПРАВЛЕНИЕМ

Д. А. КОЗЛОВСКИЙ, канд. техн. наук, доц. Т. В. МОЛОДЕЧКИНА
(Полоцкий государственный университет имени Евфросинии Полоцкой)

Представлена комбинированная приборная панель, входящая в лабораторный стенд для моделирования режимов работы стрелочных приборов. Приведены основные функции, выполняемые блоком управления лабораторного стенда, описаны особенности его конструкции. Обоснован выбор программной среды Arduino IDE для разработки программного обеспечения. Проведен выбор библиотек, использующихся для организации взаимодействия между платой Arduino и компонентами стенда. Разработано и детально рассмотрено программное обеспечение для блока управления лабораторного стенда.

Ключевые слова: *приборная панель, блок управления, программное обеспечение, среда Arduino IDE, программный код.*

Введение. В настоящее время шаговые двигатели находят широкое применение в электронике. Их используют для управления положением какого-либо объекта, для вращения рабочей части изделия с заданной скоростью или на заданный угол [1].

При решении задач, где задается положение объекта в пространстве, необходимо обеспечить два важных показателя: точность позиционирования и время выполнения команды. В таких случаях целесообразно использовать шаговые двигатели в качестве исполнительных механизмов, поскольку они имеют высокую точность позиционирования и повторяемость перемещений [2].

Точность позиционирования шагового двигателя определяется его конструктивным исполнением, увеличить ее можно за счет использования режимов полушаг и микрошаг. Быстродействие зависит от режима работы и системы управления, т. е. может задаваться схемотехническим и программным способами. Логические сигналы для управления работой шагового двигателя должны формироваться микроконтроллером [3].

В представленной статье рассмотрены особенности разработки программного обеспечения (ПО) блока управления лабораторного стенда для моделирования режимов работы стрелочных приборов.

Функции, выполняемые блоком управления лабораторного стенда. Блок управления должен получать данные о скорости движения и оборотах двигателя посредством COM-порта в течение всего времени работы лабораторного стенда. Вывод показаний тахометра и спидометра должен производиться посредством сервоприводов и указательной стрелки на панели приборов.

Особенности конструкции блока управления лабораторного стенда. Блок управления спроектирован на базе платформы Arduino. Роль программного модуля выполняет плата Arduino Nano v.3. Это обусловлено широкими возможностями платформы Arduino, гибкостью ее применения, простотой разработки и отладки программного обеспечения. На плате по умолчанию присутствует разъем USB Type Mini, через который осуществляется прошивка платы. Инициирование прошивки платы осуществляется нажатием кнопки в среде для разработки и отладки ПО.

Для вывода информации используются сервоприводы SG90[1]. Широкое использование сервоприводов связано с тем, что для них характерны стабильная работа, высокая устойчивость к помехам, малые габариты и широкий диапазон контроля скорости. Важными особенностями сервоприводов являются способность увеличивать мощность и обеспечение обратной информационной связи. Из этого следует, что при прямом направлении контур является передатчиком энергии, а при обратном – передатчиком информации, которая используется для улучшения точности управления. Подобное решение позволит также уменьшить количество задействованных выводов микроконтроллера (рисунок).

Выбор среды для разработки ПО. Разработчиком платформы Arduino в качестве среды для разработки ПО рекомендуется Arduino IDE. Программное обеспечение в этой среде представляет собой последовательность команд: программного кода на языках C и C++.

Для любой платы Arduino можно написать программный код классическим методом, т. е. в виде последовательного списка команд на языках C и C++ в среде Arduino IDE[2].

Данная среда имеет ряд достоинств: поддержка всех плат Arduino, возможность использования сторонних библиотек, что обеспечивает быстроту и удобство работы. Кроме того, для написания программного кода достаточно знать базовые возможности языков C и C++, так как именно на последнем пишется код программы в среде, а пользовательские библиотеки часто пишутся и на чистом C.

[1] Сервоприводы Ардуино SG90, MG995, MG996: схема подключения и управление [Электронный ресурс]. URL: https://arduinomaster.ru/motor-dvigatel-privod/servoprivody-arduino-sg90-mg995-shema-podklyuchenie-upravlenie/.
[2] Arduino IDE [Electronic resource]. URL: https://ru.wikipedia.org/wiki/Arduino_IDE.

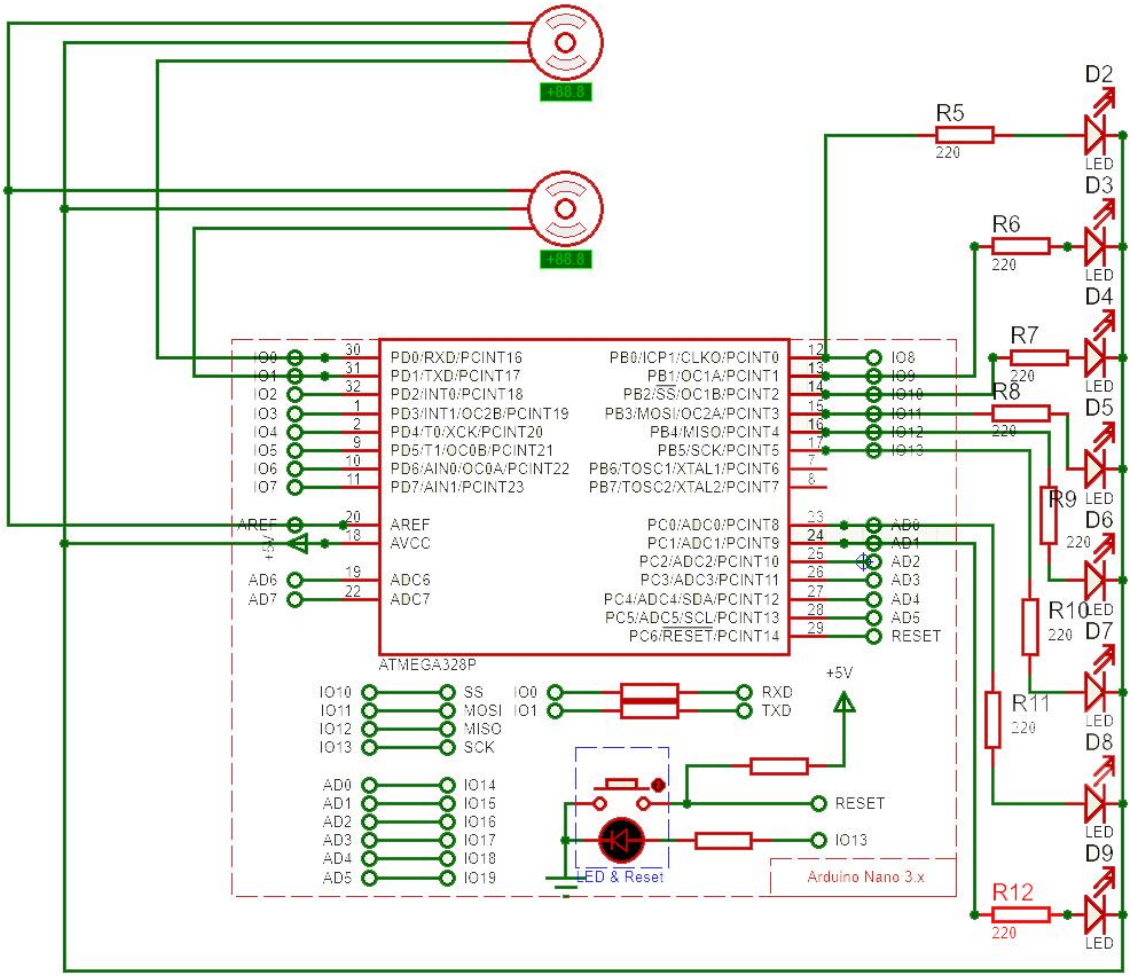

Рисунок. – Схема подключения электрорадиоэлементов в блоке управления лабораторного стенда

В качестве альтернативных сред для разработки программного обеспечения рассматривались FLProg и XOD[3].

В среде FLProg прошивка для плат Arduino создается с использованием графических языков FBD и LAD. Эти языки в области программирования микроконтроллеров являются стандартом. К достоинствам среды можно отнести следующее: автоматический контроль за использованием входов-выходов, проверку уникальности имен и согласованности типов данных, корректности проекта в целом и указание на наличие ошибок.

Недостаток среды FLProg состоит в том, что не для всех компонентов существует поддержка прямо в среде. Соответственно, нет возможности, как в Arduino IDE, для неподдерживаемых компонентов добавить стороннюю библиотеку и работать с помощью понятных и удобных команд. Такие компоненты необходимо программировать с нуля.

Среда разработки XOD, подобно FLProg, позволяет создавать программу с помощью визуальных блоков и устанавливать связи между ними, но используемый графический язык отличается от аналога. К преимуществам среды XOD можно отнести понятный и удобный интерфейс, возможность графического программирования без знания C и C++ и работы в браузере без установки. Кроме того, среда XOD распространяется бесплатно, как и Arduino IDE[4].

Подобно FLProg, в среде XOD имеется сложность программирования отдельных неподдерживаемых компонентов.

Проанализировав вышеописанные программные среды, для разработки программного обеспечения разрабатываемого устройства было решено использовать среду Arduino IDE. Это позволило применять сторонние библиотеки для связи платы с компонентами программно, в результате процесс программирования был упрощен и занял меньше времени.

[3] URL: https://flprog.ru.
[4] A visual programming language for microcontrollers [Electronic resource]. URL: https://xod.io.

Разработка ПО. При разработке конструкции блока управления были использованы электрорадиоэлементы широкого применения, поэтому для написания программы было достаточно легко найти библиотеки, представляющие собой программное обеспечение с открытым исходным кодом. Такие библиотеки могут быть использованы для некоммерческой деятельности с указанием разработчика библиотеки.

Для организации взаимодействия между платой Arduino и сервоприводами SG90 удобно использовать библиотеку Servo[5]. Библиотека Servo позволяет одновременно управлять 12 сервоприводами на большинстве плат Arduino и 48 сервоприводами на Arduino Mega. На контроллерах, отличных от Mega, использование библиотеки отключает возможность задействовать выходы 9 и 10 в режиме широтно-импульсной модуляции (ШИМ), даже если привод не подключен к этим выводам. На плате Mega могут быть использованы до 12 сервоприводов без потери функциональности ШИМ. При использовании Mega для управления 12–23 сервоприводами нельзя будет задействовать выходы 11 и 12 для ШИМ.

Для начала необходимо установить вышеупомянутые библиотеки в Arduino IDE, а затем подключить их к скетчу следующим образом:

```
#include <Servo.h>
```

Секция предварительной конфигурации и подготовки для работы с бибилиотеками

Далее необходимо определить назначение каждого из задействованных выходов Arduino.

```
const int SPEEDO_PIN          = 10;  // Сигнал спидометра
const int RPM_PIN             = 9;   // Сигнал тахометра
const int LEFT_INDICATOR      = 4;   // Левый поворотник
const int RIGHT_INDICATOR     = 5;   // Правый поворотник
const int PARKING_BRAKE       = 6;   // Ручной тормоз
const int FUEL_WARNING        = 7;   // Заканчивается топливо
const int LOW_BEAM            = 3;   // Ближний свет
const int HIGH_BEAM           = 2;   // Дальний свет
const int MOTOR_BRAKE         = 8;   // Торможение двигателем
const int BRAKE_AIR_2         = A1;  // Ошибка давления воздуха в тормозной системе
const int BRAKE_AIR_1         = A0;  // Нехватка давления воздуха в тормозной системе
const int BATTERY_LOW         = 11;  // Аккумуляторная батарея разряжена
const int OIL_TEMP            = 12;  // Слишком высокая температура масла (двигатель)
const int WATER_TEMP          = 13;  // Слишком высокая температура воды (система охлаждения)
```

Секция подготовки и инициализации

Внутрь функции «setup» помещается код для инициализации компонентов, а также вывода тестовой (стандартной) информации для проверки работоспособности устройства. Этот код будет выполнен один раз при запуске разрабатываемой установки. Функция «setup» будет реализована следующим образом:

```
void setup()
{
 Serial.begin(115200);

 // Инициализация сервоприводов
 speedo.attach(SPEEDO_PIN);
 speedo.write(180);

 rpm.attach(RPM_PIN);
 rpm.write(180);

 // Инициализация индикаторов
 pinMode(LEFT_INDICATOR, OUTPUT);
 pinMode(RIGHT_INDICATOR, OUTPUT);
 pinMode(PARKING_BRAKE, OUTPUT);
 pinMode(FUEL_WARNING, OUTPUT);
 pinMode(LOW_BEAM, OUTPUT);
 pinMode(HIGH_BEAM, OUTPUT);
 pinMode(MOTOR_BRAKE, OUTPUT);
 pinMode(BRAKE_AIR_2, OUTPUT);
 pinMode(BRAKE_AIR_1, OUTPUT);
 pinMode(BATTERY_LOW, OUTPUT);
 pinMode(OIL_TEMP, OUTPUT);
 pinMode(WATER_TEMP, OUTPUT);
```

[5] URL: https://arduino.ru/Reference/Library/Servo.

```
 // Тестирование приборной панели
 digitalWrite(LEFT_INDICATOR, 0);
 digitalWrite(RIGHT_INDICATOR, 0);
 digitalWrite(PARKING_BRAKE, 0);
 digitalWrite(FUEL_WARNING, 0);
 digitalWrite(LOW_BEAM, 0);
 digitalWrite(HIGH_BEAM, 0);
 digitalWrite(MOTOR_BRAKE, 0);
 digitalWrite(BRAKE_AIR_2, 0);
 digitalWrite(BRAKE_AIR_1, 0);
 digitalWrite(BATTERY_LOW, 0);
 digitalWrite(OIL_TEMP, 0);
 digitalWrite(WATER_TEMP, 0);

 delay(500);

 speedo.write(0);
 rpm.write(0);
 digitalWrite(LEFT_INDICATOR, 1);
 digitalWrite(RIGHT_INDICATOR, 1);
 digitalWrite(PARKING_BRAKE, 1);
 digitalWrite(FUEL_WARNING, 1);
 digitalWrite(LOW_BEAM, 1);
 digitalWrite(HIGH_BEAM, 1);
 digitalWrite(MOTOR_BRAKE, 1);
 digitalWrite(BRAKE_AIR_2, 1);
 digitalWrite(BRAKE_AIR_1, 1);
 digitalWrite(BATTERY_LOW, 1);
 digitalWrite(OIL_TEMP, 1);
 digitalWrite(WATER_TEMP, 1);

 delay(500);

 speedo.write(180);
 rpm.write(180);
 digitalWrite(LEFT_INDICATOR, 0);
 digitalWrite(RIGHT_INDICATOR, 0);
 digitalWrite(PARKING_BRAKE, 0);
 digitalWrite(FUEL_WARNING, 0);
 digitalWrite(LOW_BEAM, 0);
 digitalWrite(HIGH_BEAM, 0);
 digitalWrite(MOTOR_BRAKE, 0);
 digitalWrite(BRAKE_AIR_2, 0);
 digitalWrite(BRAKE_AIR_1, 0);
 digitalWrite(BATTERY_LOW, 0);
 digitalWrite(OIL_TEMP, 0);
 digitalWrite(WATER_TEMP, 0);

}
```

Внутрь функции «loop» помещается код основной части программы. Он будет выполняться циклически в течение всего периода работы измерительной установки. Функция «loop» будет реализована следующим образом:

```
void read_serial_byte_set_servo(Servo& servo, bool invert)
{
  serial_byte = Serial.read();
  serial_byte = (serial_byte < 0) ? 0 : ((serial_byte > 180) ? 180 : serial_byte);
  if (invert)
    servo.write(180 - serial_byte);
  else
    servo.write(serial_byte);
}

void skip_serial_byte()
{

  (void)Serial.read();
}
```

```
void digitalWriteFromBit(int port, int value, int shift)
{
 digitalWrite(port, (value   shift) & 0x01);
}

void loop()
{
 if (Serial.available() < 16)
   return;

 serial_byte = Serial.read();
 if (serial_byte != PACKET_SYNC)
   return;

 serial_byte = Serial.read();
 if (serial_byte != PACKET_VER)
 {
  return;
 }

 read_serial_byte_set_servo(speedo, SERVO_DIR_INVERT);     // Speed   telemetry.speed
 read_serial_byte_set_servo(rpm, SERVO_DIR_INVERT);        // RPM    telemetry.engine_rpm

 skip_serial_byte(); // Brake air pressure      telemetry.brake_air_pressure
 skip_serial_byte(); // Brake temperature       telemetry.brake_temperature
 skip_serial_byte(); // Fuel ratio              fuel_ratio
 skip_serial_byte(); // Oil pressure            telemetry.oil_pressure
 skip_serial_byte(); // Oil temperature         telemetry.oil_temperature
 skip_serial_byte(); // Water temperature       telemetry.water_temperature
 skip_serial_byte(); // Battery voltage         telemetry.battery_voltage

 // Байт информации о световых индикаторах грузовика, отключены незадействованные
 serial_byte = Serial.read();
//  digitalWriteFromBit(LIGHT_P,  serial_byte, 6);              // telemetry.light_parking
 digitalWriteFromBit(LEFT_INDICATOR,  serial_byte, 5);   // telemetry.light_lblinker
 digitalWriteFromBit(RIGHT_INDICATOR, serial_byte, 4);  // telemetry.light_rblinker
 digitalWriteFromBit(LOW_BEAM,  serial_byte, 3);            // telemetry.light_low_beam
 digitalWriteFromBit(HIGH_BEAM, serial_byte, 2);            // telemetry.light_high_beam
//  digitalWriteFromBit(LIGHT_B,  serial_byte, 1);              // telemetry.light_brake
//  digitalWriteFromBit(LIGHT_R,  serial_byte, 0);              // telemetry.light_reverse

 // Байт информации об индикаторах приборной панели
 serial_byte = Serial.read();
 digitalWriteFromBit(PARKING_BRAKE, serial_byte, 7);     // telemetry.parking_brake
 digitalWriteFromBit(MOTOR_BRAKE, serial_byte, 6);       // telemetry.motor_brake
 digitalWriteFromBit(BRAKE_AIR_2, serial_byte, 5);       // telemetry.brake_air_pressure_warning
 digitalWriteFromBit(BRAKE_AIR_1, serial_byte, 4);       // telemetry.brake_air_pressure_emergency
 digitalWriteFromBit(FUEL_WARNING, serial_byte, 3);      // telemetry.fuel_warning
 digitalWriteFromBit(BATTERY_LOW, serial_byte, 2);       // telemetry.battery_voltage_warning
 digitalWriteFromBit(OIL_TEMP, serial_byte, 1);          // telemetry.oil_pressure_warning
 digitalWriteFromBit(WATER_TEMP, serial_byte, 0);        // telemetry.water_temperature_warning

 // Enabled flags
 serial_byte = Serial.read();

}
```

Заключение. В статье представлено разработанное программное обеспечение для комбинированной приборной панели с сервоприводами. Был проведен сравнительный анализ и, исходя из необходимого алгоритма работы разрабатываемого устройства, для разработки программы выбрана среда Arduino IDE. Работоспособность программного обеспечения была проверена в результате моделирования и при работе реального устройства.

ЛИТЕРАТУРА

1. Емельянов А. В., Шилин А. Н. Шаговые двигатели: учеб. пособие. – Волгоград: ВолгГТУ, 2005. – 48 с.

2. Кенио Т. Шаговые двигатели и их микропроцессорные системы управления: пер. с англ. – М.: Энергоатомиздат, 1987. – 198 с.
3. Рентюк В. Шаговые двигатели и особенности их применения // Компоненты и технологии. – 2013. – № 10. – С. 71–78.

REFERENCES

1. Emel'yanov, A. V., & Shilin, A. N. (2005). *Shagovye dvigateli [Stepper motors]*. Volgograd: VolgGTU. (In Russ.).
2. Kenio, T. (1987). *Shagovye dvigateli i ikh mikroprotsessornye sistemy upravleniya Energoatomizdat [Stepper motors and their microprocessor control systems]*. Moscow: Energoatomizdat. (In Russ.).
3. Rentyuk, V. (2013). Shagovye dvigateli i osobennosti ikh primeneniya [Stepper motors and their application features] *Komponenty i tekhnologii [Components & technologies]*, (10), 71–78. (In Russ.).

Поступила 23.02.2023

SWITCH DEVICES ON STEPPER MOTORS WITH MICROCONTROLLER CONTROL

D. KOZLOVSKY, T. MOLODECHKINA
(Euphrosyne Polotskaya State University of Polotsk)

A combined instrument panel is proposed, which is included in the laboratory stand for simulating the operation mode of pointer devices. The main functions performed by the control unit of the laboratory bench are given, the design features are especially noted. The choice of the Arduino IDE software environment for software development is substantiated. The selection of the library used to combine between the Arduino board and the components of the stand has been made. Software for the laboratory control unit has been developed and studied.

Keywords: *dashboard, program code, control unit, software, Arduino IDE.*