Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Microprocessor and Interfacing

CSE2006

Digital Assignment 1

Slot: E2

Name: Kulvir Singh

Register Number: 19BCE2074

Arduino Boards Using GPIO (Motor Control)

Introduction

GPIO stands for General-Purpose Input Output. General-Purpose Input Output (GPIO) is a digital pin of an IC. It can be used as input or output for interfacing devices. If we want to read the switch's state, sensor data, etc then we need to configure it as input. And if we want to control the LED brightness, motor rotation, show text on display, etc then we need to configure it as output. It can also be explained as the following: A general-purpose input/output (GPIO) is an uncommitted digital signal pin on an integrated circuit or electronic circuit board which may be used as an input or output, or both, and is controllable by the user at runtime.

GPIOs have no predefined purpose and are unused by default. If used, the purpose and behavior of a GPIO is defined and implemented by the designer of higher assembly-level circuitry: the circuit board designer in the case of integrated circuit GPIOs, or system integrator in the case of board-level GPIOs.

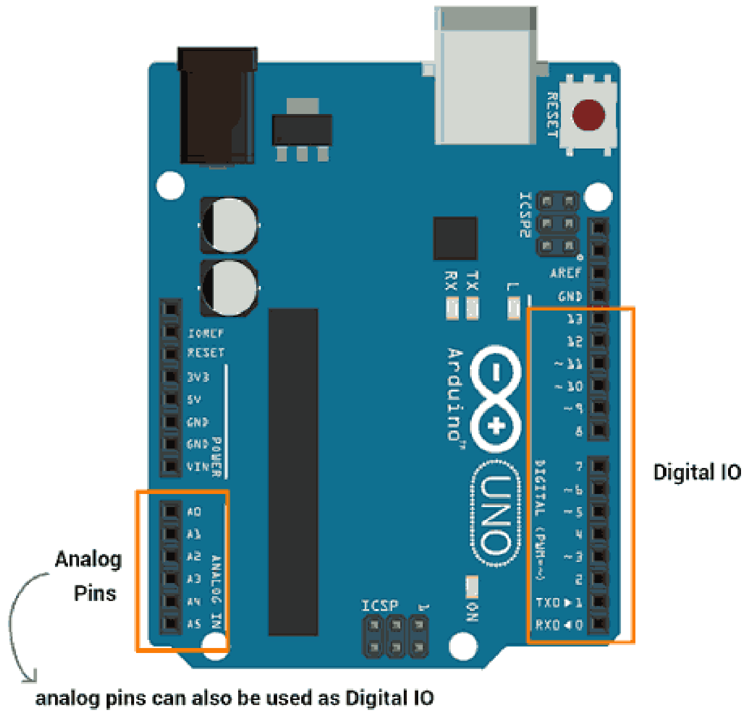


Fig 1 : The Arduino based GPIO board.

Arduino GPIO Pins

The Arduino Uno board has various digital IO pins which can be used for input/output devices. Figure 1 shows the digital IO pins of Arduino Uno. Arduino analog pins can also be used as digital input/output pins.

Digital Output

Arduino (ATmega) digital pins can be configured as output to drive output devices. We have to configure these pins to use as output.
To configure these pins, pinMode () function is used which set direction of pin as input or output.
pinMode(pin no, Mode)
This function is used to configure GPIO pin as input or output.

pin no number of pin whose mode we want to set.

Mode INPUT, OUTPUT or INPUT_PULLUP

E.g. pinMode (3, OUTPUT) //set pin 3 as output

These Arduino (ATmega) pins can source or sink current up to 40 mA which is sufficient to drive led, LCD display but not sufficient for motors, relays, etc.

Note: While connecting devices to Arduino output pins use resistor. If any connected device to Arduino withdraw current more than 40 mA from the Arduino then it will damage the Arduino pin or IC.

These pin produce output in terms of HIGH (5 V or 3.3 V) or LOW (0 V). We can set output on these pins using digitalWrite () function.

digitalWrite (pin no, Output value)
This function is used to set output as HIGH (5 V) or LOW (0 V)

pin no number of a pin whose mode we want to set.

Output value HIGH or LOW

E.g. digitalWrite (3, HIGH)

Digital Input

To read data from senor or from any device/circuit, we need to configure digital pin as input. Arduino pin are set as digital input (default). So, there is no need to configure pin as input.

To configure pin as digital input, pinMode () function is used. We can read data from GPIO pin using digitalRead() function.

digitalRead(pin)
It is used to read data from specified GPIO pin.

Digital Input with Pull-up Resistor

Sometimes switching between one state to another or pins configured as input with nothing connected to them may arise situation of High-impedance state i.e. floating state. This state may report random changes in pin state.
To avoid this state, there is option of adding pull-up (to +5V) or pull-down (to Gnd) resistor which helps to set the input to a known state. Following image show the high impedance (undefine) state and pull-up resistor.
Pull-up resistor concept

Pull-up resistor
Arduino (ATmega) has in-built configurable pull-up resistor. These resistors enable using pinMode () with mode set to INPUT_PULLUP. When connecting a device or sensor to a pin configured as input with pull-up, the other end should be connected to ground.
e.g. pinMode (3, INPUT_PULLUP).

We can configure input pull-up in another way too. If we set direction of pin as input and then write HIGH value on that pin will turn on the pull-up resistor. In other manner, if we write HIGH on pin configured as OUTPUT and then configure that pin as input will also enable the pull-up resistor.

```
e.g. pinMode (3, INPUT)        //set pin as input

digitalWrite (3, HIGH)          //setting high on input pin enables pullup
```

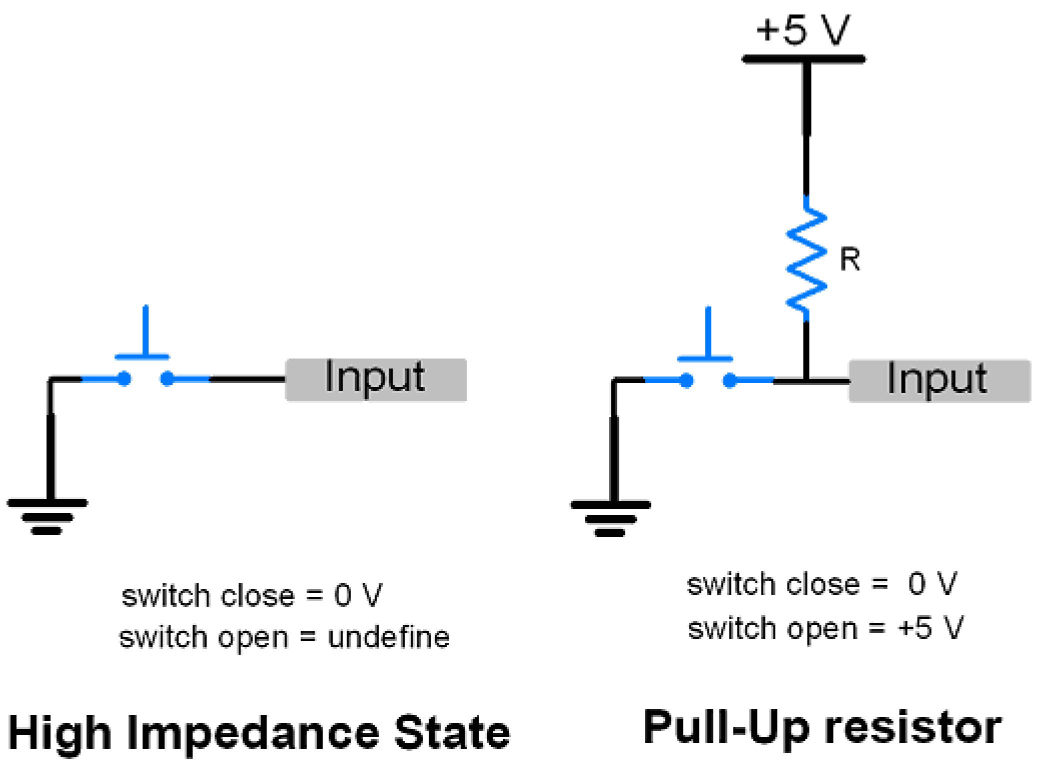


Fig. 2. The pull-up resistor.

Implementation of the GPIO

GPIO interfaces vary widely. In some cases, they are simple—a group of pins that can switch as a group to either input or output. In others, each pin can be set up to accept or source different logic voltages, with configurable drive strengths and pull ups/downs. Input and output voltages are usually, but not always, limited to the supply voltage of the device with the GPIOs, and may be damaged by greater voltages.

A GPIO pin's state may be exposed to the software developer through one of a number of different interfaces, such as a memory-mapped I/O peripheral, or through dedicated IO port instructions. Some GPIOs have 5 V tolerant inputs: even when the device has a low supply voltage (such as 2 V), the device can accept 5 V without damage.

A GPIO port is a group of GPIO pins (usually 8 GPIO pins) arranged in a group and controlled as a group.

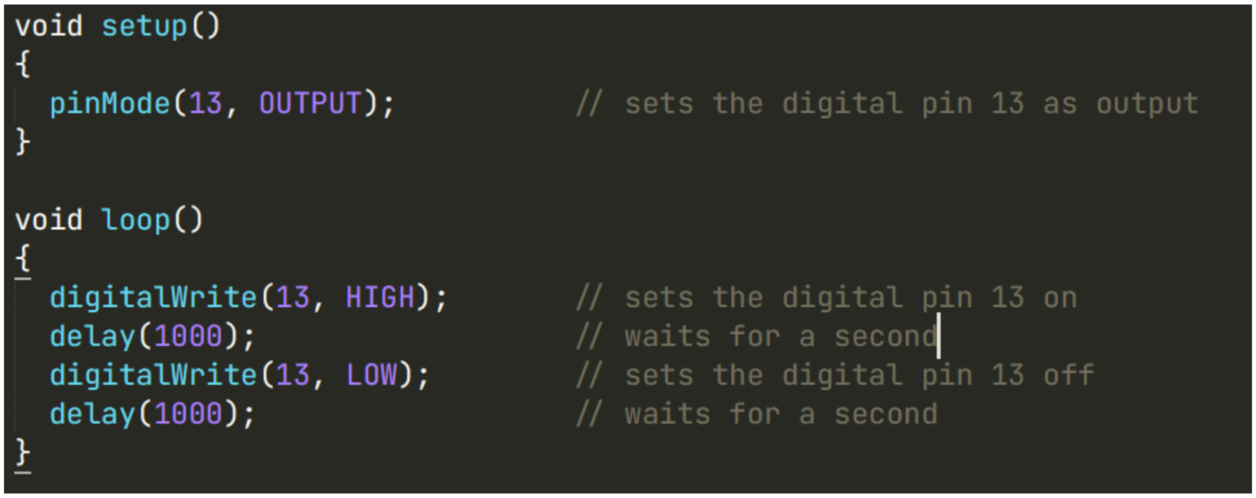
GPIO abilities may include:

GPIO pins can be configured to be input or output
GPIO pins can be enabled/disabled
Input values are readable (usually high or low)
Output values are writable/readable
Input values can often be used as IRQs (usually for wakeup events)

Some code based example of GPIO usages

Blink LED Using Arduino

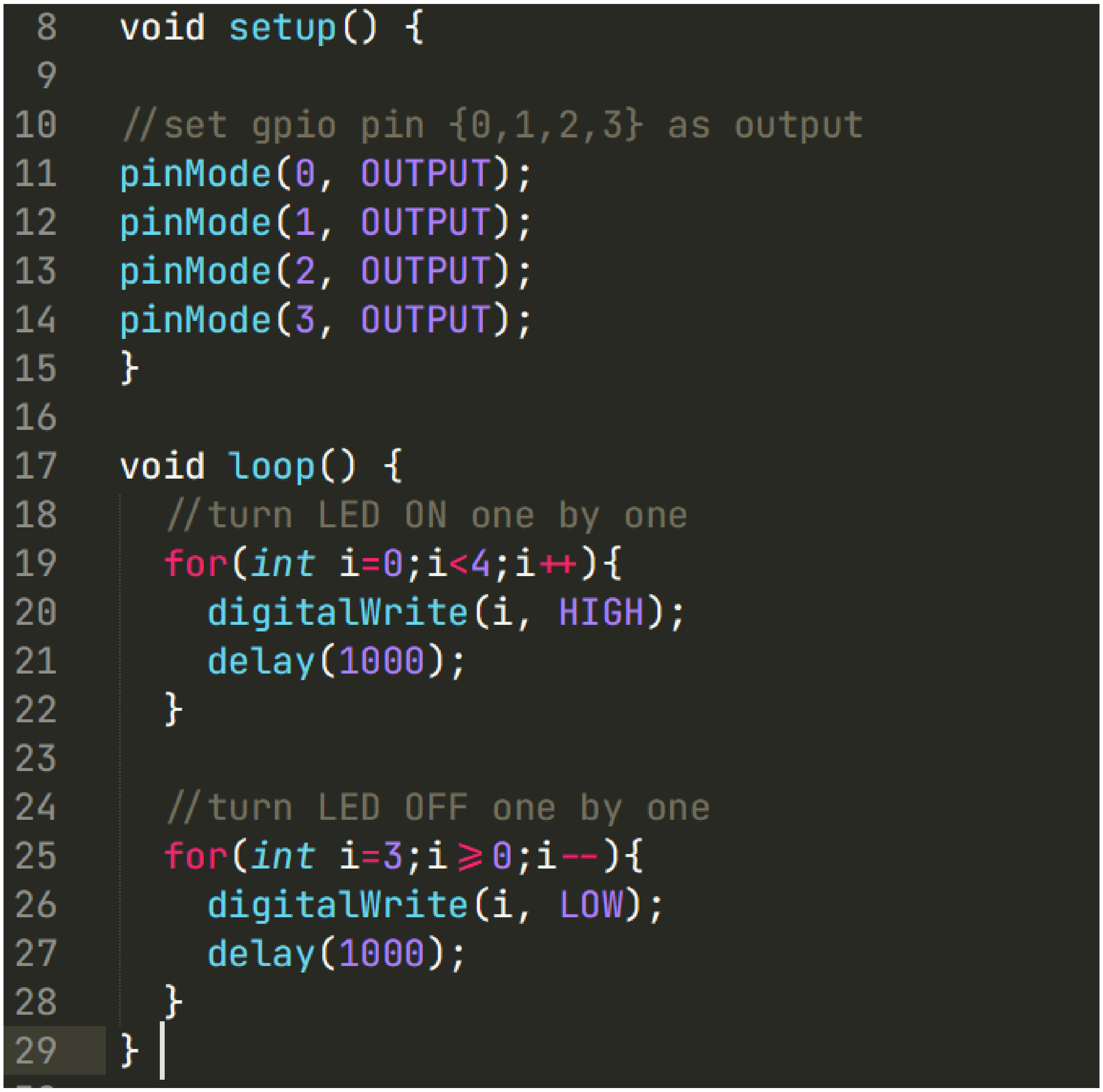
Let’s write a program for led blinking using Arduino Uno. Here, we will blink the on-board LED of Arduino Uno which is connected to digital pin 13.



```
void setup()
{
  pinMode(13, OUTPUT);          // sets the digital pin 13 as output
}

void loop()
{
  digitalWrite(13, HIGH);       // sets the digital pin 13 on
  delay(1000);                  // waits for a second
  digitalWrite(13, LOW);        // sets the digital pin 13 off
  delay(1000);                  // waits for a second
}
```

Sequentially Turning LED ON-OF using Arduino

Let’s turn ON-OFF led sequentially. Here, first we will turn four LED ON one by one which means LED1 will ON first then LED1 and LED2 will ON and so on. When at last all four leds will get ON then turn them off in reverse sequence one by one. That means turn LED4 OFF then turn LED3 OFF and so on. Four LED’s are connected to pin 0 – 3 of Arduino Uno. While uploading program to Arduino just remove the connection of pin 0 and 1 as they are RXD and TXD pin which are used by serial communication for program uploading. When program uploaded successfully, then connect the connection of pin 0 and 1.

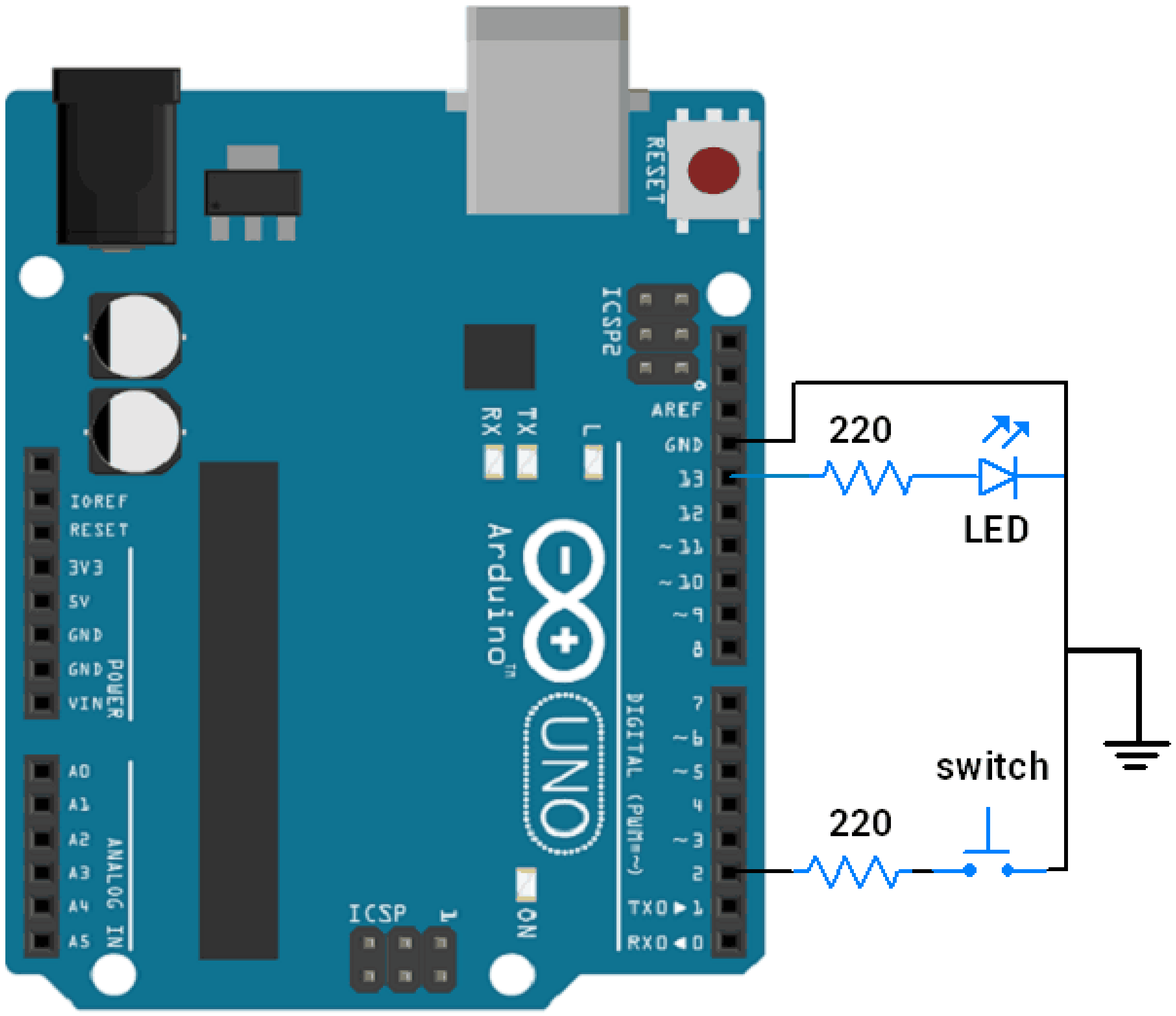


```
void setup() {

//set gpio pin {0,1,2,3} as output
pinMode(0, OUTPUT);
pinMode(1, OUTPUT);
pinMode(2, OUTPUT);
pinMode(3, OUTPUT);
}

void loop() {
  //turn LED ON one by one
  for(int i=0;i<4;i++){
    digitalWrite(i, HIGH);
    delay(1000);
  }

  //turn LED OFF one by one
  for(int i=3;i>=0;i--){
    digitalWrite(i, LOW);
    delay(1000);
  }
}
```

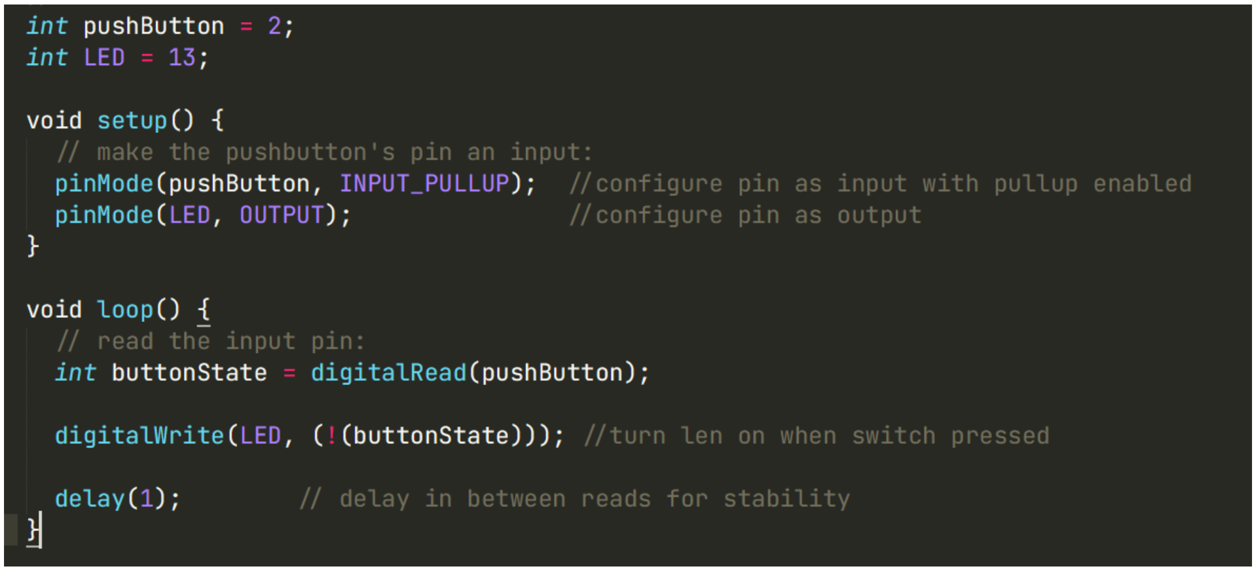
Control LED via Switch using Arduino

Let's control LED ON-OFF using switch interfaced to Arduino Uno.

Interfacing Diagram



Code :



```
int pushButton = 2;
int LED = 13;

void setup() {
  // make the pushbutton's pin an input:
  pinMode(pushButton, INPUT_PULLUP);  //configure pin as input with pullup enabled
  pinMode(LED, OUTPUT);               //configure pin as output
}

void loop() {
  // read the input pin:
  int buttonState = digitalRead(pushButton);

  digitalWrite(LED, (!(buttonState))); //turn len on when switch pressed

  delay(1);        // delay in between reads for stability
}
```

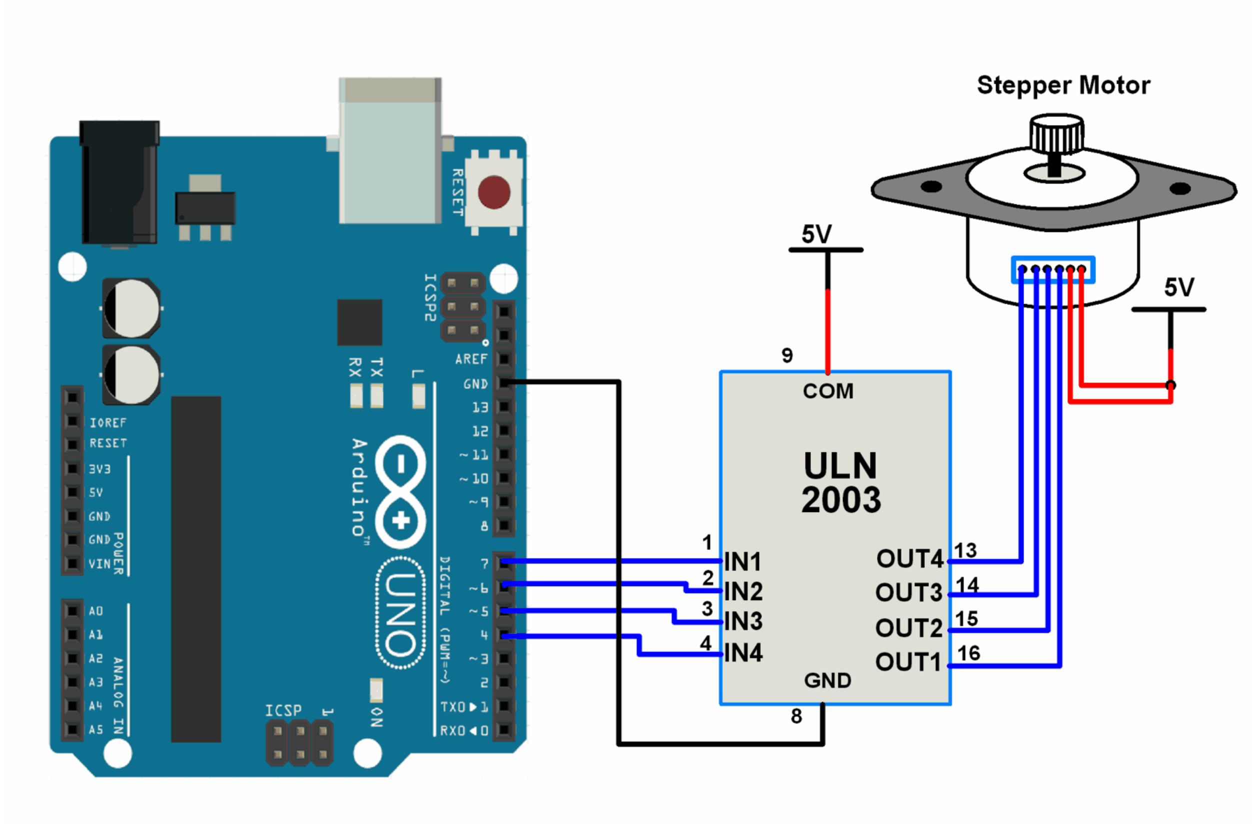
Motor Control Using Arduino

In this section we will be discussing examples where GPIO arduino can be used to achieve motor control.

Stepper Motor Interfacing with Arduino UNO

Stepper motor is a brushless DC motor that divides the full rotation angle of 360° into a number of equal steps. The motor is rotated by applying a certain sequence of control signals. The speed of rotation can be changed by changing the rate at which the control signals are applied.

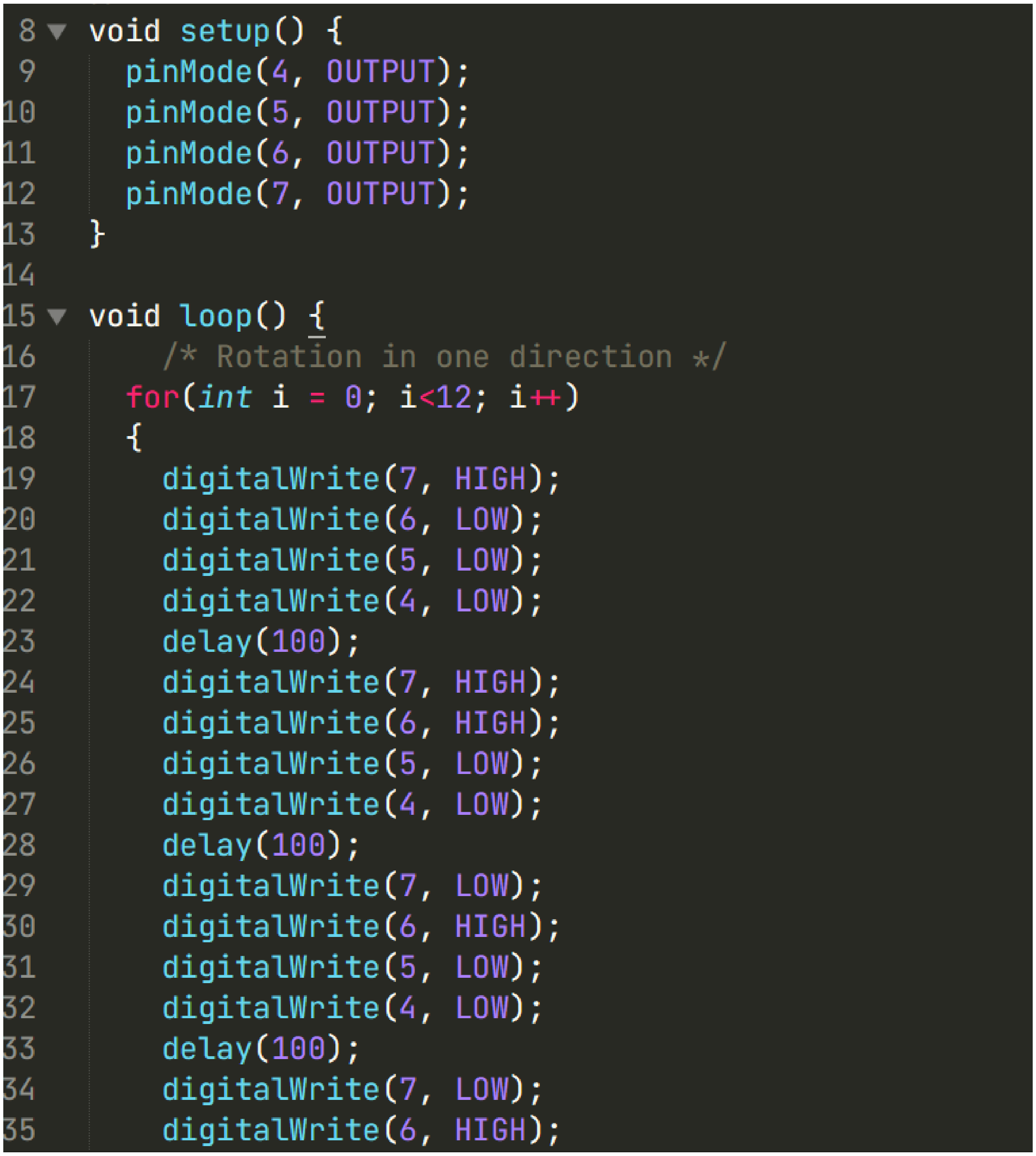
Interfacing diagram



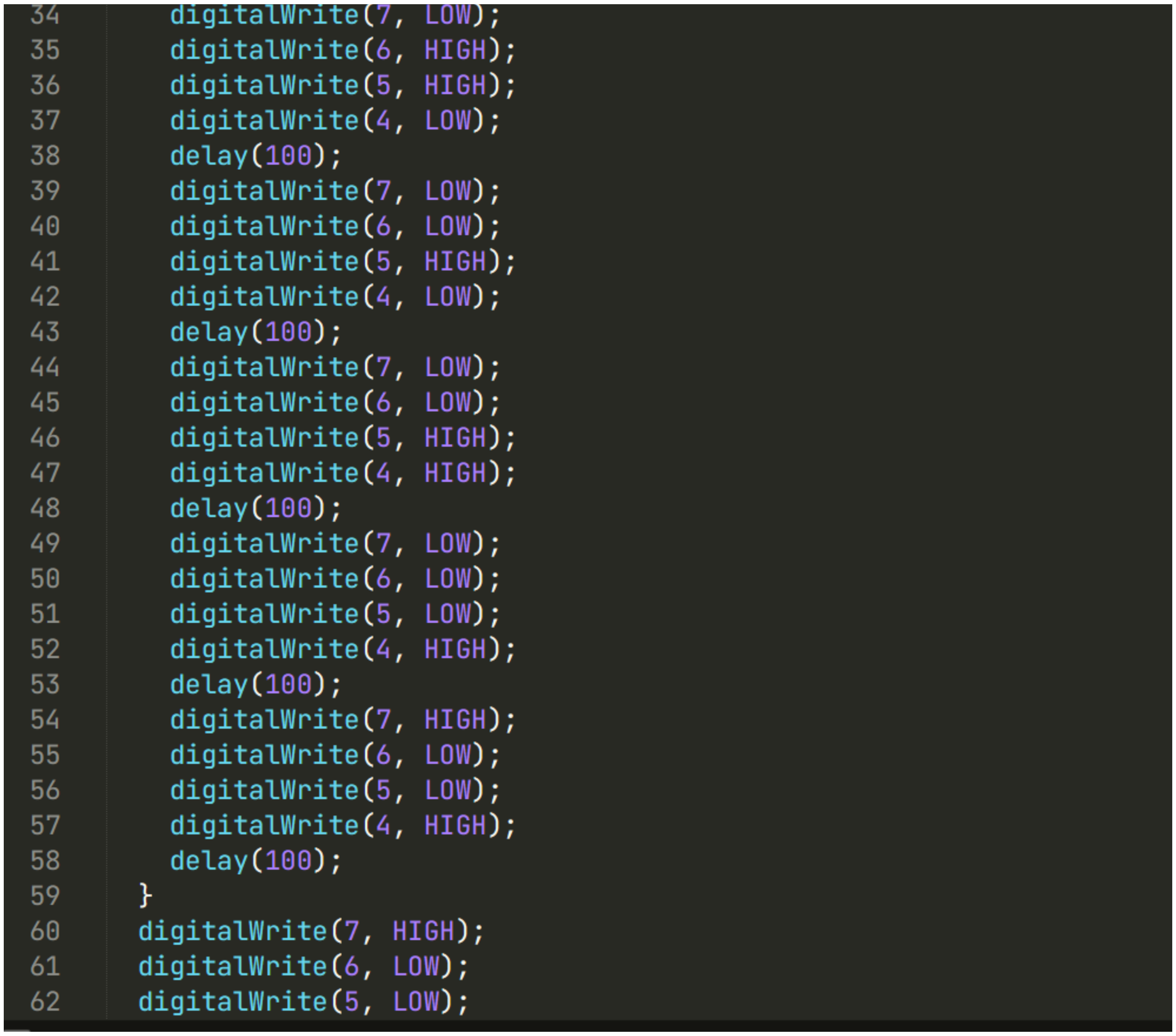
Use case :
Rotating stepper motor in clockwise and counter clockwise directions alternately. Here, we are using six wire unipolar stepper motor. Only four wires are required to control this stepper motor. The two center tap wires of the stepper motor are connected to 5V supply.
ULN2003 driver is used to drive the stepper motor.
To find winding coils and their center tap leads, measure resistance in between the leads. From center leads we will get half the resistance value as compared to the resistance between winding ends.

Code :

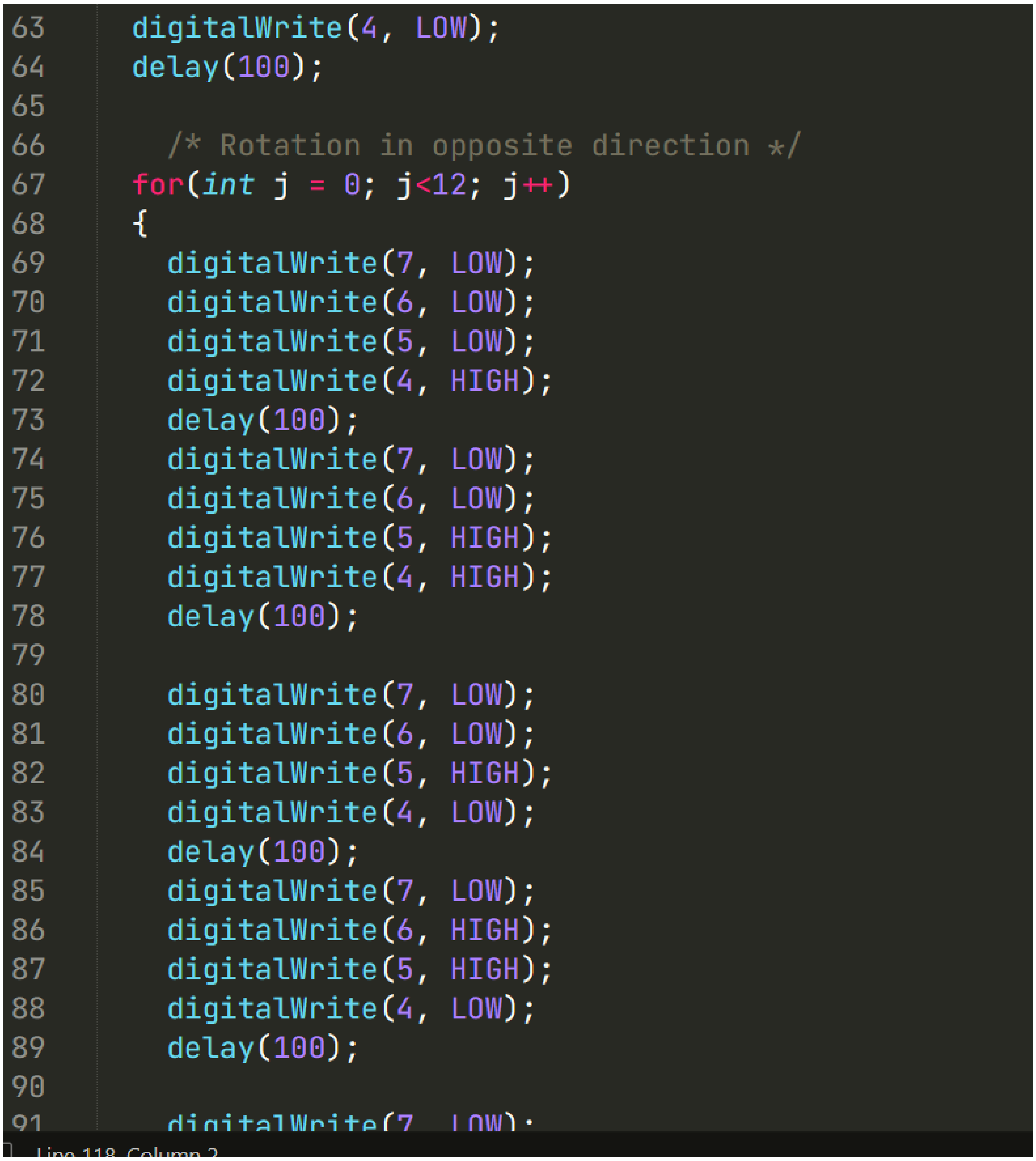
As seen in the above code, it is visible that the GPIO has been configured to take input and output. Initially the pins are configured as output high and are giving the output. It is also performing a digital write command hence we can clearly state that a GPIO has been used.



```
void setup() {
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
}

void loop() {
    /* Rotation in one direction */
  for(int i = 0; i<12; i++)
  {
    digitalWrite(7, HIGH);
    digitalWrite(6, LOW);
    digitalWrite(5, LOW);
    digitalWrite(4, LOW);
    delay(100);
    digitalWrite(7, HIGH);
    digitalWrite(6, HIGH);
    digitalWrite(5, LOW);
    digitalWrite(4, LOW);
    delay(100);
    digitalWrite(7, LOW);
    digitalWrite(6, HIGH);
    digitalWrite(5, LOW);
    digitalWrite(4, LOW);
    delay(100);
    digitalWrite(7, LOW);
    digitalWrite(6, HIGH);
```

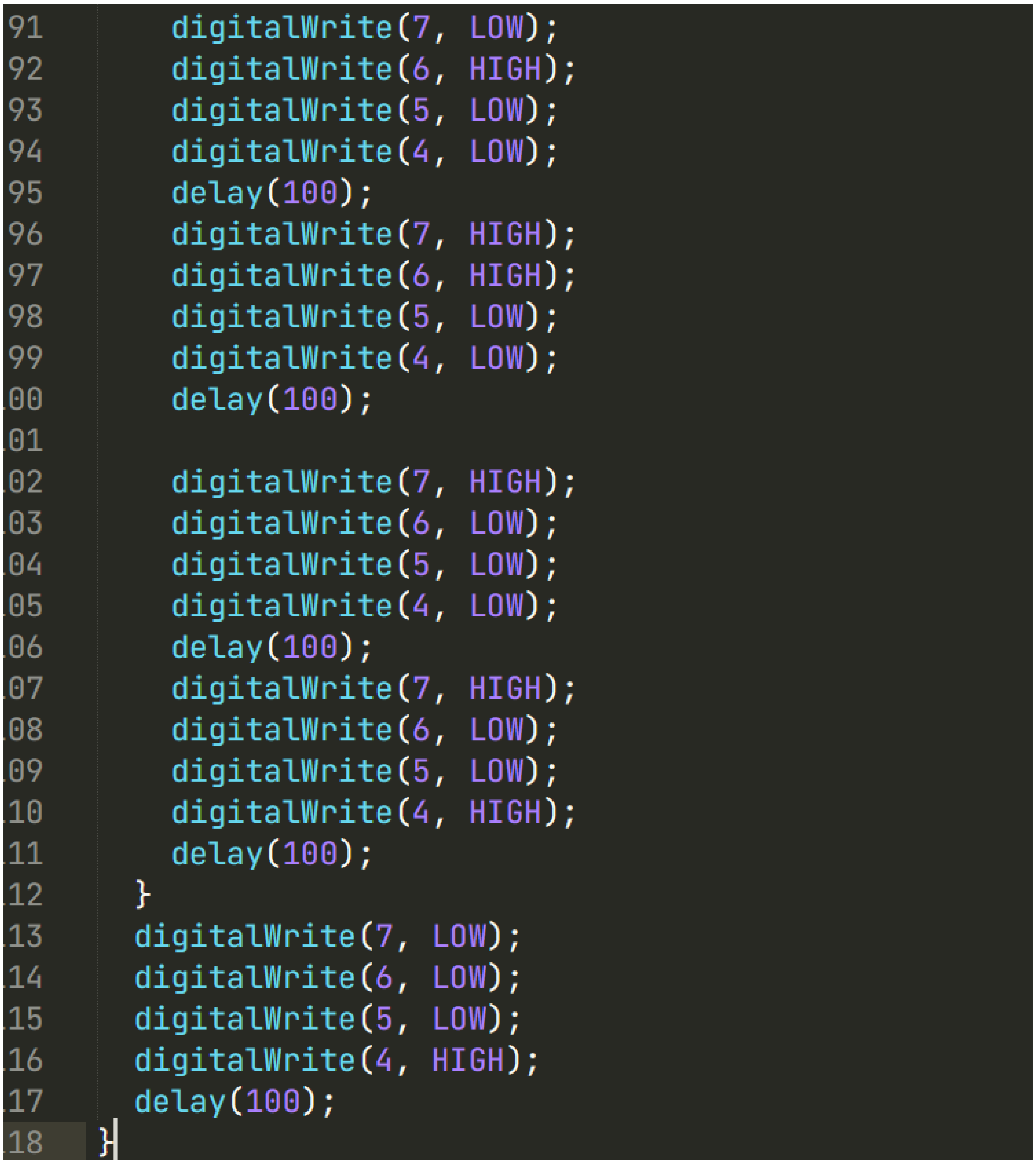


```
    digitalWrite(7, LOW);
    digitalWrite(6, HIGH);
    digitalWrite(5, HIGH);
    digitalWrite(4, LOW);
    delay(100);
    digitalWrite(7, LOW);
    digitalWrite(6, LOW);
    digitalWrite(5, HIGH);
    digitalWrite(4, LOW);
    delay(100);
    digitalWrite(7, LOW);
    digitalWrite(6, LOW);
    digitalWrite(5, HIGH);
    digitalWrite(4, HIGH);
    delay(100);
    digitalWrite(7, LOW);
    digitalWrite(6, LOW);
    digitalWrite(5, LOW);
    digitalWrite(4, HIGH);
    delay(100);
    digitalWrite(7, HIGH);
    digitalWrite(6, LOW);
    digitalWrite(5, LOW);
    digitalWrite(4, HIGH);
    delay(100);
  }
  digitalWrite(7, HIGH);
  digitalWrite(6, LOW);
  digitalWrite(5, LOW);
```



```
  digitalWrite(4, LOW);
  delay(100);

    /* Rotation in opposite direction */
  for(int j = 0; j<12; j++)
  {
    digitalWrite(7, LOW);
    digitalWrite(6, LOW);
    digitalWrite(5, LOW);
    digitalWrite(4, HIGH);
    delay(100);
    digitalWrite(7, LOW);
    digitalWrite(6, LOW);
    digitalWrite(5, HIGH);
    digitalWrite(4, HIGH);
    delay(100);

    digitalWrite(7, LOW);
    digitalWrite(6, LOW);
    digitalWrite(5, HIGH);
    digitalWrite(4, LOW);
    delay(100);
    digitalWrite(7, LOW);
    digitalWrite(6, HIGH);
    digitalWrite(5, HIGH);
    digitalWrite(4, LOW);
    delay(100);

    digitalWrite(7, LOW);
```

Line 118, Column 2



```
    digitalWrite(7, LOW);
    digitalWrite(6, HIGH);
    digitalWrite(5, LOW);
    digitalWrite(4, LOW);
    delay(100);
    digitalWrite(7, HIGH);
    digitalWrite(6, HIGH);
    digitalWrite(5, LOW);
    digitalWrite(4, LOW);
    delay(100);

    digitalWrite(7, HIGH);
    digitalWrite(6, LOW);
    digitalWrite(5, LOW);
    digitalWrite(4, LOW);
    delay(100);
    digitalWrite(7, HIGH);
    digitalWrite(6, LOW);
    digitalWrite(5, LOW);
    digitalWrite(4, HIGH);
    delay(100);
  }
  digitalWrite(7, LOW);
  digitalWrite(6, LOW);
  digitalWrite(5, LOW);
  digitalWrite(4, HIGH);
  delay(100);
}
```

<u>Conclusion :</u>

To conclude, this assignment has brought about the implementation, usage of GPIO using arduino systems. Apart from that, it also showcases the basics of the general-purpose input output organization and interfacing of the arduino board. The application of the same is made clear with regard to motor control.