Introduction to the ATmega328P Microcontroller (Arduino Uno)

1.0 Learning Outcomes....1
2.0 Introduction to Microcontrollers....2
3.0 Microcontrollers in a Control System....4
4.0 Digital Outputs....6
5.0 PWM Outputs....6
6.0 Summary....7
Recommended Citation....8

1.0 Learning Outcomes

After studying this ACE-CORE guide, you should be able to demonstrate the Learning Outcomes listed in Table 1. On completion of the guide, revisit Table 1 and confirm whether each Learning Outcome has been achieved.

Table 1: *Learning Outcomes from the Introduction to the ATmega328P Microcontroller Guide*

ID	Learning Outcome	Achieved
1.	Identify and describe the key hardware components of the Arduino Uno, including its power supply options, digital and analogue I/O pins, PWM-capable pins and memory.	☐
2.	Explain the role of a microcontroller within a control system, and map its inputs and outputs onto the control system block diagram (controller, actuator, process and measurement).	☐
3.	Distinguish between open-loop and closed-loop control, and explain the limitations of open-loop actuation without measurement and feedback.	☐
4.	Explain how digital output pins provide binary (ON/OFF) actuation, and the current limits that apply to them.	☐
5.	Explain the principles of pulse-width modulation (PWM), including duty cycle, resolution and average output voltage, and calculate the average voltage for a given PWM value.	☐
6.	Describe how Simulink and the Arduino Support Package generate C++ code from a block diagram model for deployment to a microcontroller.	☐

2.0 Introduction to Microcontrollers

A microcontroller is a small computer on a single chip, containing a processor, memory and input/output (I/O) pins. In a control system, the microcontroller runs the control algorithm: it reads signals from sensors through its input pins, calculates what the system should do, and sends commands to actuators through its output pins.

This guide uses the Arduino Uno R3, a low-cost microcontroller board built around the ATmega328P microcontroller, shown in Figure 1. The numbered components are described in Table 2.

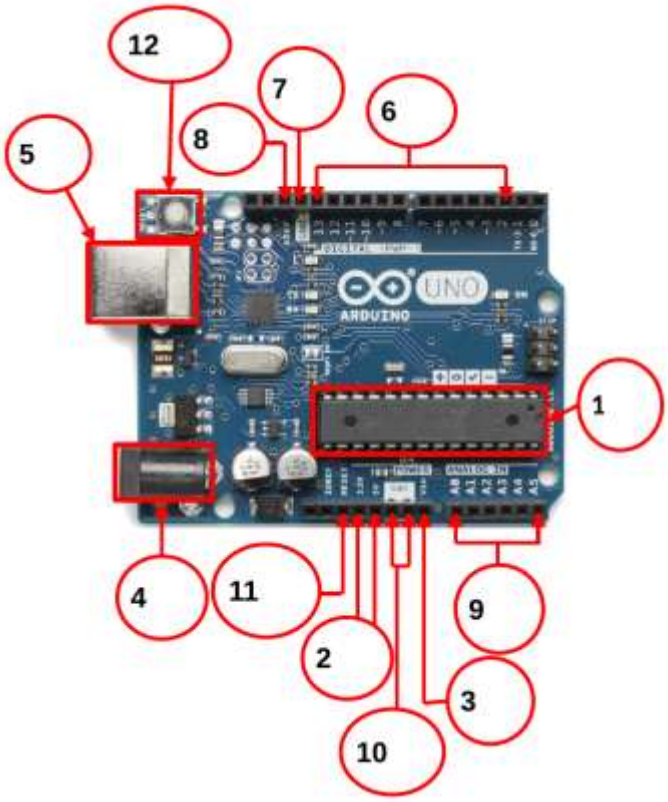


Figure 1: *Key Components of the Arduino Uno R3*

Table 2: *Hardware Overview of the Arduino Uno R3*

ID	Component	Description
1	Microcontroller	ATmega328P, which runs the program and manages all I/O
2	3.3 V and 5 V supply pins	Regulated outputs for powering sensors and modules
3	Voltage in (VIN) pin	External supply input, 7–12 V recommended
4	DC barrel jack	External supply input, 7–12 V recommended
5	USB Type-B connector	Power and programming from a computer
6	Digital I/O pins	14 pins (D0–D13), six of which support PWM
7	Ground (GND)	Common ground for the digital pins
8	Analogue reference (AREF)	Optional external reference voltage for the ADC
9	Analogue input pins	Six pins (A0–A5) connected to the 10-bit ADC
10	Ground pins (GND)	Common ground on the power header
11	Reset pin	Restarts the microcontroller when pulled LOW
12	Reset button	Restarts the microcontroller manually

The ATmega328P on the Arduino Uno operates at 5 V. The board also provides a regulated 3.3 V supply pin for sensors that need a lower voltage. Power can be supplied through the USB connector, which is convenient for programming and low-power use, or through an external DC supply connected to the barrel jack or the VIN pin for standalone operation.

The board provides 14 digital I/O pins. Six of these (D3, D5, D6, D9, D10 and D11, marked with a ~ on the board) support pulse-width modulation (PWM) for controlling devices such as LEDs and motors. Pins D0 and D1 are also used for serial communication with the computer, so they are best avoided for other purposes. Six analogue input pins (A0–A5) read sensor voltages, and the AREF pin allows an external reference voltage to be used. Several ground (GND) pins provide a common electrical reference, and a reset pin and reset button restart the program.

The main operating and electrical characteristics are listed in Table 3. The recommended input voltage is 7–12 V, with absolute limits of 6–20 V. Each I/O pin can source or sink up to 40 mA as an absolute maximum; in practice, 20 mA or less per pin is recommended for reliable operation. The program is stored in 32 KB of flash memory, variables are held in 2 KB of static random-access memory (SRAM) while the program runs, and 1 KB of EEPROM stores data that must be kept when the power is removed. The 16 MHz clock provides enough processing speed for real-time control.

Table 3: *Operating and Electrical Characteristics*

Property	Value
Operating voltage	5 V
Input voltage (recommended)	7–12 V
Input voltage (limits)	6–20 V
DC current per I/O pin	40 mA absolute maximum (20 mA recommended)
DC current from the 3.3 V pin	50 mA
Flash memory	32 KB (0.5 KB used by the bootloader)
Static random-access memory (SRAM)	2 KB
Electrically erasable programmable read-only memory (EEPROM)	1 KB
Clock speed	16 MHz

3.0 Microcontrollers in a Control System

Linking Microcontroller I/O to the Block Diagram

A control system is commonly represented as a block diagram made up of a controller, an actuator, a process and a measurement device (sensor) in the feedback path. Figure 2 shows where the microcontroller fits:

- The control algorithm (for example on–off, proportional or PID control) runs as software on the ATmega328P.
- The control output, u, is applied to the actuator through the output pins: digital pins for ON/OFF actions, or PWM pins for continuous-valued actions.
- The measured output from the sensor returns to the microcontroller through its input pins, as covered in the Microcontroller-Based Measurement guide.

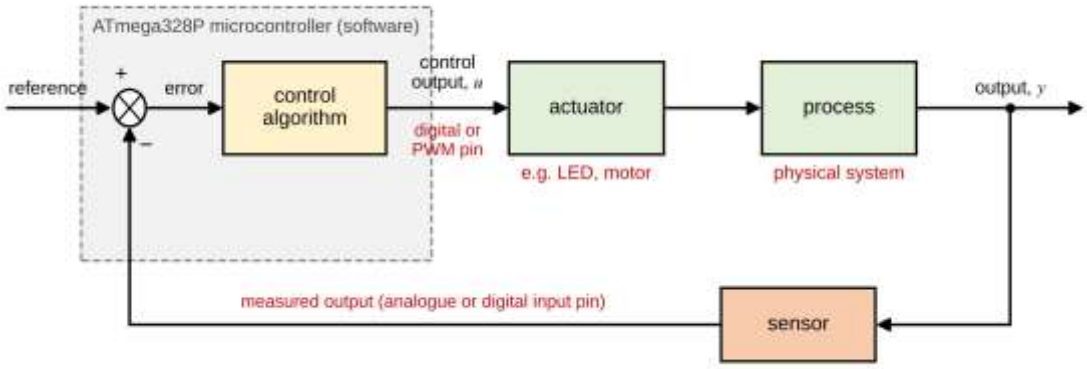


Figure 2: *How a Microcontroller Implements a Closed-Loop Control System*

This guide focuses on the output side. To begin with, the simpler open-loop arrangement in Figure 3 is considered. The algorithm generates a control input, u, and applies it directly to the system, with no measurement fed back and no error calculated. Disturbances are neglected for now. The limitation of open-loop operation is that the microcontroller has no way of knowing whether the system has actually done what was intended, for example whether a motor has reached the required speed under load. Measurement and feedback, introduced in the guides that follow, remove this limitation.

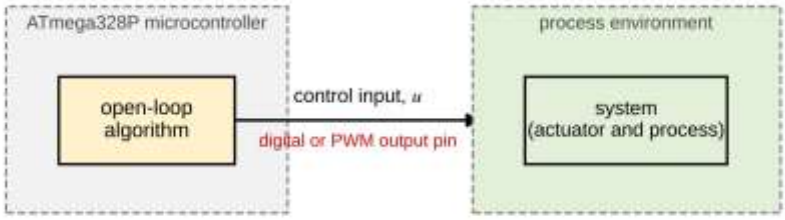


Figure 3: *Open-Loop Algorithm and System Considered in this Guide*

Simulink Code Generation for the Arduino

Real-time control algorithms are commonly written in C++, because it is efficient and well suited to embedded hardware. Simulink supports a model-based design approach, in which the control algorithm is built as a block diagram that closely mirrors the control system representation in Figure 2.

Using the Simulink Arduino Support Package, the block diagram is automatically converted into C++ code and uploaded to the Arduino. The support package provides ready-made blocks for inputs (digital inputs and analogue inputs) and outputs (digital outputs and PWM), so the focus is on the control logic rather than low-level microcontroller programming. The advantages of this workflow include:

- rapid prototyping: an algorithm can be changed and redeployed quickly
- consistency: the same model is used for simulation and for the hardware
- platform independence: the same model can be deployed to different hardware

- integrated verification: results from simulation and hardware can be compared directly

4.0 Digital Outputs

A digital output pin drives one of two logic levels: LOW (0 V) or HIGH (5 V on the Arduino Uno). Digital outputs are used to switch devices such as LEDs, relays and motor drivers, or to send logic signals to other digital circuits. When a pin is set as an output, it actively sources current (when HIGH) or sinks current (when LOW). The load must therefore stay within the current limits in Table 3: for example, an LED needs a series resistor, and a motor must be driven through a motor driver rather than directly from the pin.

In a control system, a digital output gives a binary (ON/OFF) actuation: the actuator is either fully on or fully off. This is the basis of on–off control, such as a thermostat that switches a heater on below a set temperature and off above it.

5.0 PWM Outputs

Pulse-width modulation (PWM) allows a digital pin to deliver a controllable *average* voltage. The pin is switched rapidly between HIGH and LOW at a fixed frequency (about 490 Hz on most Arduino Uno PWM pins, and about 980 Hz on D5 and D6). The proportion of each period for which the signal is HIGH is called the duty cycle. Increasing the duty cycle increases the average voltage, and so the average power delivered to the load:

$$V_{avg} = D \times 5\ \text{V} \tag{1}$$

where D is the duty cycle, expressed as a fraction between 0 and 1. Because most actuators, such as DC motors, heaters and LEDs (as perceived by the eye), respond much more slowly than the switching, they respond to the average value. PWM therefore provides a simple and efficient way to achieve continuous-valued control using purely digital hardware, without a dedicated analogue output.

The precision of PWM depends on its resolution, which is the number of available duty-cycle levels. The Arduino Uno PWM outputs have 8-bit resolution, so the duty cycle is set by a value from 0 to 255 (256 levels). The average voltage is then:

$$V_{avg} = \frac{\text{value}}{255} \times 5\ \text{V} \tag{2}$$

giving a voltage step of $5/255 \approx 0.02$ V per level. Figure 4 shows PWM signals for duty cycles of 0%, 25%, 50%, 75% and 100%, together with their average voltages. At 0% the signal is always LOW and no power is delivered. At 25% the signal is HIGH for a quarter of each period, giving a low average voltage of about 1.25 V. At 50% and 75% the average rises to about 2.5 V and 3.75 V. At 100% the signal is always HIGH and the maximum power is

delivered. Although the signal is only ever fully LOW or fully HIGH, changing the duty cycle controls the average voltage, and so the power, seen by the actuator.

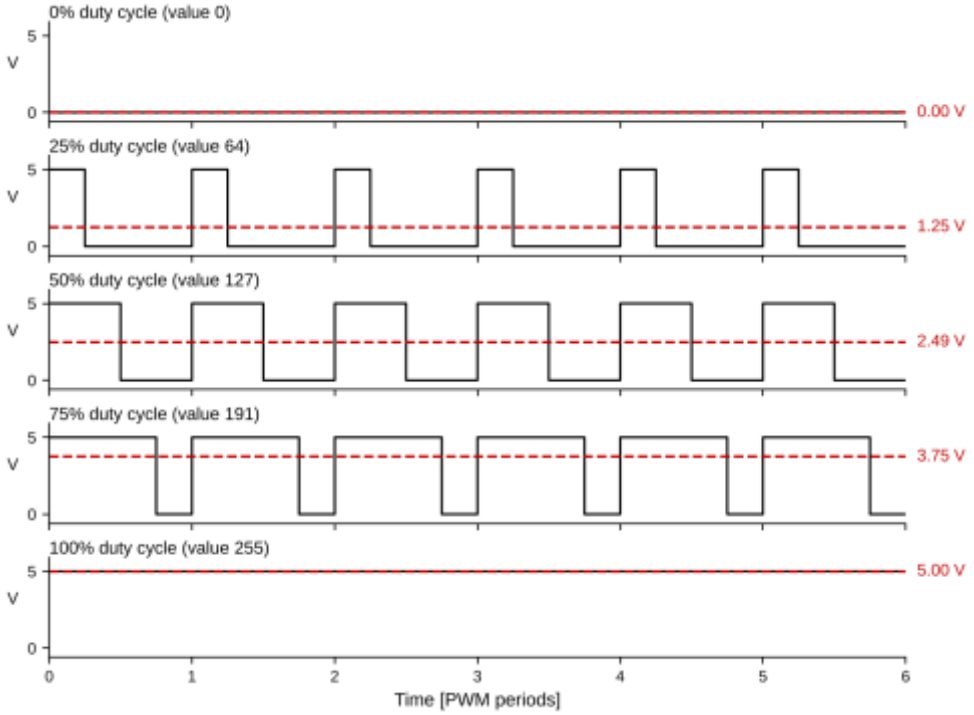


Figure 4: *PWM Signals at a Fixed Switching Frequency for Several Duty Cycles, with their Average Voltages*

6.0 Summary

This guide introduced the Arduino Uno R3 and its ATmega328P microcontroller as a practical platform for implementing control systems. Its key hardware features and electrical limits were described, and its inputs and outputs were mapped onto the control system block diagram: the control algorithm runs in software, actuators are driven from the output pins, and measurements return through the input pins.

Two ways of driving an actuator were introduced. Digital outputs give binary ON/OFF actuation, and PWM outputs give a continuous-valued average voltage set by the duty cycle, with 256 levels on the Arduino Uno. Simulink's automatic code generation provides a direct link from a control algorithm designed in simulation to its real-time implementation. The open-loop arrangement considered here cannot correct for disturbances; the following guides introduce measurement, feedback and closed-loop control.

Recommended Citation

If you have used this guide in your teaching, learning or research, please cite it using the reference below, adapting the format as required by your preferred referencing style.

Pickering, J. E. (2026). Introduction to the ATmega328P Microcontroller (Arduino Uno). ACE-Lab. ACE-CORE: Operate (O), Version 1.0. https://zenodo.org/records/23326158