CHAPTER 16 Programming Fundamentals

Learning Outcomes

- Introduction to Programming
- Pseudo Code
- Microcontroller/SBC
- Embedded C Programming

Programming means giving instructions to a robot or computer to do certain jobs. In robots, programming tells the robot how to move, how to sense things, and how to make decisions. Without programming, a robot is just a machine with no brain.

First, we will see how to write **algorithms-** simple step-by-step instructions to solve a problem. Then, we will use **flowcharts**, which are pictures that show these steps in an easy way. You will also learn about **pseudo code**, which is like writing a program in simple English before using a real programming language.

Programming Fundamentals

Programming means giving instructions to a robot or computer so it can do tasks on its own. In robots, programming helps them move, sense things, and make decisions.

1. Algorithms

An **algorithm** is a list of steps to solve a problem or do a task. It's like a recipe the robot follows. For Example: Steps to move a robot forward for 5 seconds:

1. Turn on the motor.
2. Move forward.
3. Wait for 5 seconds.
4. Stop the motor.

What is an Algorithm?

An **algorithm** is a set of clear and simple steps that tell you how to solve a problem or complete a task. It's like a recipe or instructions you follow to get a result. In robotics, algorithms are used to guide the robot on what to do and how to do it.

Why Are Algorithms Important?

- They help break down a problem into small, easy steps.

- They make programming easier because you know exactly what to tell the robot.
- They help find the best or fastest way to complete a task.

Example of an Algorithm

Imagine you want a robot to pick up an object and place it somewhere else. An algorithm for this task could be:

1. Move the robot arm to the object.
2. Close the gripper to hold the object.
3. Lift the arm up.
4. Move the arm to the new location.
5. Open the gripper to release the object.
6. Move the arm back to the starting position.

Key Points

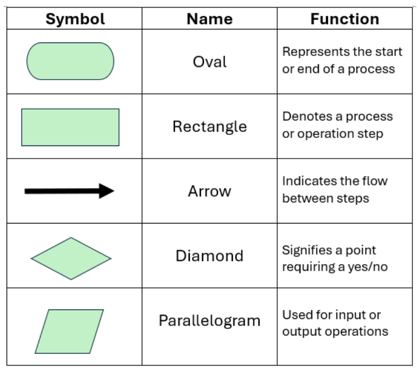
- Algorithms are written in simple steps anyone can follow.
- They are the first step in writing a computer program.
- Good algorithms are clear, simple, and easy to follow.

2. Flowcharts

A **flowchart** is a picture that shows the steps of an algorithm using shapes and arrows.

- **Oval** means start or end.
- **Rectangle** means an action or instruction.
- **Diamond** means a question or decision (like yes or no).
- **Arrow** shows the direction to the next step.

Flowcharts make it easy to understand how a program works.



Symbol	Name	Function
⬭	Oval	Represents the start or end of a process
▭	Rectangle	Denotes a process or operation step
→	Arrow	Indicates the flow between steps
◇	Diamond	Signifies a point requiring a yes/no
▱	Parallelogram	Used for input or output operations

What is a Flowchart?

A **flowchart** is a picture or diagram that shows the steps of a process or algorithm using different shapes and arrows. It helps us understand how a program or task works by showing the order of actions clearly.

Why Are Flowcharts Useful?

- They make it easy to see what happens first, next, and last.
- They help programmers plan and organize their ideas before writing code.
- They make it simple to explain how a program works to others.

Common Flowchart Symbols

- **Oval**: Shows the **Start** or **End** of a process.
- **Rectangle**: Represents a **Process** or action to be done.
- **Diamond**: Represents a **Decision** (a yes/no question).
- **Arrow**: Shows the **flow** or direction from one step to the next.

How to Create a Flowchart

1. Identify the steps in the process.
2. Use the right symbols for each step.
3. Connect the symbols with arrows to show the flow.
4. Make sure the flowchart starts at the start and ends at the end.

3. Pseudo Code

Pseudo code is writing the steps of a program using simple English words. It helps us plan the program before writing real code. Example:

Turn on an LED if a switch is ON:

```
IF switch is ON THEN
  Turn LED ON
ELSE
  Turn LED OFF
END IF
```

Why Use Pseudo Code?

- It helps you plan your program clearly before writing real code.
- It makes it easier to think about the logic without worrying about the exact syntax.
- It's useful for sharing your ideas with others who may not know programming languages.

How is Pseudo Code Written?

- Write instructions step-by-step like a list.
- Use simple words to describe actions and decisions.
- Use keywords like IF, ELSE, WHILE, FOR to show decisions and loops.
- Don't worry about punctuation or exact spelling like in real code.

4. Microcontroller / SBC Programming

Robots have tiny computers called **microcontrollers** or **single board computers (SBCs)** inside them. Programming these computers means writing instructions to control the robot's parts like sensors and motors.

Microcontrollers use languages like **Embedded C**, and SBCs like Raspberry Pi can use languages like Python or C. In robotics, the "brain" of a robot is usually a **microcontroller** or a **single-board computer (SBC)**. These small computers are programmed to control sensors, motors, lights, and more. Without programming, these devices can't do anything on their own.

What is a Microcontroller?

A **microcontroller** is a small computer built into a single chip. It contains:

- A processor (to think)
- Memory (to remember instructions)
- Input/output pins (to talk to sensors and motors)

Examples: Arduino Uno, ATmega328, PIC microcontrollers
Microcontrollers are used in simple robots, toys, washing machines, traffic lights, etc.

What is a Single Board Computer (SBC)?

An **SBC** is a small computer built on a single circuit board. It can run full operating systems like Linux. **Examples:** Raspberry Pi, NVIDIA Jetson Nano
SBCs are more powerful than microcontrollers. They can do complex tasks like computer vision, face detection, internet control, and more.

Programming Microcontrollers

Microcontrollers are programmed using **Embedded C**, a version of the C language that talks directly to the hardware.

What we can control with microcontroller programming:

- Read data from sensors (temperature, light, distance)
- Turn motors on/off
- Blink LEDs
- Respond to button presses
- Count time or create delays

Example: Turn on an LED using Embedded C (Arduino)

```
void setup()
{
  pinMode(13, OUTPUT); // Set pin 13 as output
}
void loop()
{
  digitalWrite(13, HIGH); // Turn LED ON
  delay(1000);            // Wait for 1 second
  digitalWrite(13, LOW);  // Turn LED OFF
  delay(1000);            // Wait for 1 second
}
```

Programming SBCs

SBCs can be programmed using Python, C, or other high-level languages. Programming an SBC is like programming a desktop computer, but it also lets you control hardware.

What SBC programming can do:

- Control cameras and display images
- Connect to Wi-Fi or internet
- Run artificial intelligence programs
- Do speech recognition or object tracking
- Store and analyse sensor data

Difference between a Microcontroller and an SBC

Feature	Microcontroller	SBC (e.g., Raspberry Pi)
Cost	Low	Higher
Speed	Slower	Faster
Power use	Very low	Higher
Programming	Embedded C	Python / Linux OS
Best for	Simple control tasks	Complex processing tasks

Programming Examples for Microcontrollers (Arduino – Embedded C)

Example 1: Blinking an LED

```
void setup() {
  pinMode(13, OUTPUT);  // Set pin 13 as output
}
void loop() {
  digitalWrite(13, HIGH); // Turn LED ON
  delay(500);             // Wait 0.5 seconds
  digitalWrite(13, LOW);  // Turn LED OFF
  delay(500);             // Wait 0.5 seconds
}
```

Example 2: Reading a Button Press

```
int buttonPin = 2;
int ledPin = 13;
int buttonState = 0;
void setup() {
  pinMode(buttonPin, INPUT);  // Set pin 2 as input
  pinMode(ledPin, OUTPUT);    // Set pin 13 as output
}
void loop() {
  buttonState = digitalRead(buttonPin);  // Read button status
  if (buttonState == HIGH) {
    digitalWrite(ledPin, HIGH);         // Turn ON LED
  } else {
    digitalWrite(ledPin, LOW);          // Turn OFF LED
  }
}
```

Example 3: Controlling a Motor

```
int motorPin = 9;
void setup() {
  pinMode(motorPin, OUTPUT);
}
```

```
void loop() {
  analogWrite(motorPin, 128); // Run motor at 50% speed (range 0-255)
}
```

Example 4: Using an Ultrasonic Sensor to Measure Distance

```
const int trigPin = 9;
const int echoPin = 10;
long duration;
int distance;
void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(9600);
}
void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2;
  Serial.print("Distance: ");
  Serial.println(distance);
  delay(500);
}
```

Programming Examples for SBCs (Raspberry Pi – Python)

Example 1: Blink an LED

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
while True:
    GPIO.output(18, GPIO.HIGH)
    time.sleep(1)
    GPIO.output(18, GPIO.LOW)
    time.sleep(1)
```

Example 2: Reading a Button

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.IN, pull_up_down=GPIO.PUD_UP)
while True:
    input_state = GPIO.input(18)
    if input_state == False:
        print("Button Pressed")
        time.sleep(0.2)
```

Example 3: Controlling a Motor Using GPIO

```
import RPi.GPIO as GPIO
import time
motorPin = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(motorPin, GPIO.OUT)
GPIO.output(motorPin, True)  # Motor ON
time.sleep(5)
GPIO.output(motorPin, False) # Motor OFF
GPIO.cleanup()
```

Basic Embedded C Programming

Embedded C is a programming language used to write code for microcontrollers. It lets you control the robot’s hardware directly.

Turn an LED ON using Embedded C:

```
#include <avr/io.h>
int main(void) {
    DDRB |= (1 << PB0);  // Set pin PB0 as output
    PORTB |= (1 << PB0); // Turn ON LED connected to PB0
    while(1) {
        // Keep the LED ON
    }
    return 0;
}
```

What is Embedded C Programming?

Embedded C is a version of the C programming language used to write software that directly controls **hardware devices**, such as microcontrollers (e.g., Arduino, PIC, AVR). It is called "*embedded*" because the program is *embedded* into a small chip to make machines like robots, washing machines, remote controls, and more work.

Why Use Embedded C in Robotics?

In robotics, we use microcontrollers to:

- Control motors
- Read sensors
- Turn LEDs on/off
- Make decisions and run logic

All this is done by writing programs using Embedded C.

Structure of an Embedded C Program

Most Embedded C programs have two main parts:

1. setup() function: This runs once when the system is powered ON. It is used to set up pins, sensors, or motors.

2. loop() function: This runs again and again forever. It keeps checking conditions and taking actions.

Blink an LED

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

void loop() {
  digitalWrite(13, HIGH); // Turn LED ON
  delay(1000);            // Wait for 1 second
  digitalWrite(13, LOW);  // Turn LED OFF
  delay(1000);            // Wait for 1 second
}
```

Explanation:

- pinMode() sets pin 13 as output (to send signals).
- digitalWrite(HIGH/LOW) turns pin ON or OFF.
- delay() waits for a certain time in milliseconds.

Common Functions in Embedded C (Arduino)

Function	Purpose
pinMode(pin, mode)	Set pin as input or output
digitalWrite(pin, HIGH/LOW)	Turn a pin ON or OFF
digitalRead(pin)	Read if a pin is ON or OFF
analogRead(pin)	Read analog value (0–1023)
analogWrite(pin, value)	Output analog value (PWM)
delay(time)	Pause for time in milliseconds
Serial.begin(9600)	Start serial communication
Serial.print() / Serial.println()	Print to serial monitor

Real-World Use in Robotics

- **LED Blinking**: Robot status indicators
- **Button Reading**: Start/stop robot
- **Motor Control**: Moving wheels or arms
- **Sensor Reading**: Obstacle detection
- **Timers & Delays**: Action timing, blinking lights
- **Interrupts**: React quickly to events like bump sensors

Interrupts

An **interrupt** is like an alarm that tells the microcontroller to stop what it is doing and pay attention to something important immediately. This helps the robot react quickly, for example, when it detects an obstacle. When an interrupt happens, the microcontroller runs a special small program called an **Interrupt Service Routine (ISR)** and then goes back to the main program. In programming, interrupts are used when something **urgent** happens, like:

- A button is pressed
- A sensor detects something suddenly
- A timer goes off

Instead of constantly checking for the event (which wastes time), the microcontroller **waits** for the event and only responds when needed.

Why Are Interrupts Important in Robotics?

Robots often deal with real-time events. Using interrupts allows the robot to be faster and more responsive. For example:

- **Obstacle detection**: Stop immediately when a bump sensor is hit
- **Motor control**: Act when encoder signal changes
- **Timer tasks**: Perform a task at exact time intervals

How Interrupts Work

1. The microcontroller is running the main program.
2. An interrupt signal comes from a **pin**, **sensor**, or **timer**.
3. The microcontroller **stops** the main task.
4. It runs a special mini-program called an **Interrupt Service Routine (ISR)**.
5. Once the ISR is complete, it **returns** to the main program.

Example (Arduino – Button Interrupt)

This program turns on an LED when a button is pressed using an interrupt.

```
const int buttonPin = 2;
const int ledPin = 13;
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(buttonPin), turnOnLED, FALLING);
}
```

```
void loop() {
  // Main program does nothing here
}

void turnOnLED() {
  digitalWrite(ledPin, HIGH);  // LED turns ON when interrupt occurs
}
```

Explanation:

- attachInterrupt() sets up the interrupt.
- When the button is pressed (FALLING edge), it runs turnOnLED().
- The main loop keeps running separately.

Types of Interrupts

Type	Triggered By
External Interrupt	Button, sensor, etc.
Timer Interrupt	After a time delay
Pin Change Interrupt	When any digital pin changes

Interrupt Service Routine (ISR)

- It is a **short** function that runs when an interrupt occurs.
- It should be **quick and simple** — no delay(), no Serial.print()
- Example: Turning on a flag, setting a value, etc.

Do’s and Don’ts in an ISR

Do:

- Keep it short
- Set a flag (use a variable to signal main program)
- Change a pin value

Don’t:

- Use delay()
- Use Serial.print()
- Perform heavy processing

Advantages of Using Interrupts

- Saves processor time
- Immediate response to real-world events
- Better for multitasking and timing-critical applications

Timers

Timers help the microcontroller keep track of time. They count tiny clock pulses and are used to:

- Make delays (wait for some time)

- Measure how long something happens
- Control things like blinking lights or motor speed

What Are Timers?

A **timer** in a microcontroller is like a built-in stopwatch. It keeps counting up (or down) based on clock signals and helps the microcontroller measure **time** or perform tasks **at specific time intervals**. Timers are extremely useful in robotics for:

- Controlling the speed of motors
- Creating delays without using delay()
- Measuring time between events
- Generating signals like PWM (Pulse Width Modulation)

Why Timers Are Important in Robotics

Timers make all this possible **without slowing down** the robot's other functions. In a robot, you often need actions to happen *at the right time*, such as:

- **Turning a motor ON for exactly 2 seconds**
- **Measuring how long it takes for an ultrasonic pulse to bounce back**
- **Blinking an LED every 1 second without stopping other tasks**
- **Creating precise PWM signals to control servo motors**

How Timers Work

Every microcontroller (like an Arduino) has **one or more timers**. These timers:

- Are hardware features built into the chip
- Count at a speed defined by the system clock (e.g., 16 million counts per second for Arduino Uno)
- Can trigger an **interrupt** or set a flag when a certain count is reached

Types of Timers

Type	Description
Delay Timer	Used to wait for a certain amount of time
Interval Timer	Repeats a task at regular intervals
PWM Timer	Used to create Pulse Width Modulation
Capture Timer	Measures time between external signals

Example (Arduino): Blinking an LED Using millis() Timer

```
const int ledPin = 13;
unsigned long previousTime = 0;
const long interval = 1000; // 1 second
void setup() {
  pinMode(ledPin, OUTPUT);
}
void loop() {
  unsigned long currentTime = millis();
  if (currentTime - previousTime >= interval) {
    previousTime = currentTime;
    digitalWrite(ledPin, !digitalRead(ledPin)); // Toggle LED
  }
}
```

- **millis()** is a built-in Arduino function that returns how many milliseconds have passed since the board was powered on.

PWM Using Timer (Arduino)

Pulse Width Modulation (PWM) is generated using timers to control motor speed or LED brightness:

```
void setup() {
  pinMode(9, OUTPUT);
}
void loop() {
  analogWrite(9, 128); // 50% duty cycle (range 0–255)
}
```

Real-World Applications of Timers in Robotics

- **Motor Speed Control**: Using PWM to adjust how fast a motor runs
- **Obstacle Detection**: Using micros() to measure ultrasonic pulse delay
- **LED Indicators**: Blinking status LEDs without freezing other operations
- **Precise Task Scheduling**: Doing one task every 100 ms and another every 1 sec

Built-In Timer Functions in Arduino

Function	Purpose
millis()	Time in milliseconds since program start
micros()	Time in microseconds
delay()	Pauses the program (not recommended for multitasking)
delayMicroseconds()	Short pauses (for precise timing)

Chapter Highlights

- Programming gives instructions to robots to perform tasks automatically.
- It controls robot actions like movement, sensing, and decision-making.
- Algorithms are step-by-step instructions to solve a problem.
- They help in planning and writing efficient code.
- A visual way to represent algorithms using symbols.
- Common symbols:
    - **Oval** = Start/End
    - **Rectangle** = Process
    - **Diamond** = Decision
    - **Arrow** = Flow direction
- Flowcharts make understanding and planning easier.
- **Microcontrollers**: Small computers on chips (e.g., Arduino), programmed using **Embedded C**.
- **SBCs (Single Board Computers)**: More powerful boards (e.g., Raspberry Pi), programmed using **Python** or **C**.
- Used to control hardware like motors, LEDs, and sensors.

Exercise

Multiple-Choice Questions (MCQs)

1. What is an algorithm?
   a) A type of robot
   b) A programming language
   c) A step-by-step solution to a problem
   d) A flowchart symbol
2. Which shape in a flowchart represents a decision?
   a) Rectangle
   b) Oval
   c) Diamond
   d) Arrow
3. Pseudo code is:
   a) A real programming language
   b) A diagram
   c) Written in binary
   d) Written in simple English
4. Which programming language is used with Arduino microcontrollers?
   a) Python
   b) Java

    c) Embedded C
    d) HTML
5. An interrupt in microcontroller programming is used to:
    a) Delay the program
    b) Perform only one task
    c) React to important events immediately
    d) Generate random numbers
6. What is the function of digitalWrite(13, HIGH) in Arduino?
    a) Turns OFF LED
    b) Blinks the LED
    c) Turns ON LED
    d) Reads the sensor
7. A single-board computer like Raspberry Pi uses which language most commonly?
    a) Java
    b) HTML
    c) Python
    d) Assembly
8. What does the loop() function do in Arduino?
    a) Runs once
    b) Defines variables
    c) Runs repeatedly
    d) Uploads code
9. The function of timers is to:
    a) Store sensor data
    b) Handle internet communication
    c) Control time-based tasks
    d) Increase power supply
10. Which of the following is NOT a flowchart symbol?
    a) Arrow
    b) Triangle
    c) Oval
    d) Diamond

True or False

1. Algorithms help us write robot programs more clearly.
2. Flowcharts are written using programming languages.
3. SBCs are less powerful than microcontrollers.
4. Pseudo code helps in planning before actual coding.
5. Timers cannot be used to control motor speed.

Fill in the Blanks

1. A ______ is a step-by-step method to solve a problem.
2. A ______ is a small computer built into a chip.
3. The symbol used for a decision in a flowchart is a ______.

4. Embedded C is commonly used to program ______.
5. ______ are used to measure time or create delays in robotics.

Assertion and Reason

1. **Assertion (A):** Algorithms are helpful in planning robotic tasks.
   Reason (R): They are written in complex programming code.
   a) Both A and R are true, and R is the correct explanation of A
   b) Both A and R are true, but R is not the correct explanation of A
   c) A is true but R is false
   d) A is false but R is true
2. **Assertion (A):** Flowcharts make understanding algorithms easier.
   Reason (R): Flowcharts use visual symbols to represent steps.
   a) Both A and R are true, and R is the correct explanation of A
   b) Both A and R are true, but R is not the correct explanation of A
   c) A is true but R is false
   d) A is false but R is true
3. **Assertion (A):** Pseudo code must follow strict programming syntax.
   Reason (R): Pseudo code is an early step before writing real code.
   a) Both A and R are true, and R is the correct explanation of A
   b) Both A and R are true, but R is not the correct explanation of A
   c) A is true but R is false
   d) A is false but R is true
4. **Assertion (A):** Interrupts are used in robotics to respond to events quickly.
   Reason (R): They allow the microcontroller to ignore the event.
   a) Both A and R are true, and R is the correct explanation of A
   b) Both A and R are true, but R is not the correct explanation of A
   c) A is true but R is false
   d) A is false but R is true
5. **Assertion (A):** Timers help perform tasks after a certain delay.
   Reason (R): They are used to store data permanently.
   a) Both A and R are true, and R is the correct explanation of A
   b) Both A and R are true, but R is not the correct explanation of A
   c) A is true but R is false
   d) A is false but R is true

Short Answer Questions

1. What is the purpose of pseudo code in programming?
2. Write two advantages of using flowcharts.
3. What does the setup() function do in an Arduino program?
4. How do interrupts improve the functioning of a robot?
5. Name two programming languages used for microcontroller and SBC programming.

Long Answer Type Questions

1. Explain the differences between microcontrollers and single board computers with examples.
2. Describe the role of timers in robotics with a simple Arduino example.
3. Write a pseudo code and flowchart to turn on a motor when a button is pressed.
4. What are interrupts in microcontrollers? How do they work with an example?
5. Explain the structure of an Embedded C program using Arduino with an example.