

These are some extra assignments for basic coding, and coding for the Arduino. There are references, too.

# Arduino Coding Lessons

Basic Lessons

Peter Vaughan

---

Contents

Objective ..... 2

Assignment 1..... 3

Assignment 2..... 5

Assignment 3..... 6

Assignment 4..... 7

Project 1 ..... 9

Project 2 ..... 10

Project 3 ..... 12

Resources ..... 15

# Objective

The purpose of this expanded assignment collection is to build upon my earlier document, “Public General Use – Mockup of Class Assignments.” That initial resource provided a strong foundation for teaching general programming and introductory Arduino development.

With this new collection, I aim to explore a wider variety of assignments that address different learning goals and conceptual areas. These include fundamental programming principles, memory management, and hands-on coding with Arduino. The goal is to offer assignments that are adaptable to varying levels of experience and interest—whether you're a complete beginner or someone looking to strengthen your skills in programming or embedded systems.

These materials are designed to support continuous learning. I always encourage others to pursue what they are passionate about. For me, that passion lies in coding and embedded systems. I would love to connect with others who share similar interests—because learning and growth are always better when shared.

These assignments do not include language requirements, rubrics or submission requirements. Sample rubrics and example submissions can be found in my class mockup. Final details are left to the discretion of teachers, instructors, or coaches.

# Assignment 1

## Build a Simple Interactive Program

The objective is to learn the fundamentals of coding by building a text-based program that interacts with the user.

---

✅ **Learning Outcomes:** By the end of this assignment, students will be able to:

- Use **variables** to store data.
  - Get **input** from users and display **output**.
  - Use **conditional statements** (if/else).
  - Use **loops** to repeat actions.
  - Write and call **functions**.
- 

### **Part 1: Hello, World!**

Write a program that prints. Example:

Hello, World!

Welcome to the world of coding.

---

### **Part 2: User Interaction**

Prompt the user to enter their **name** and **age**, then display a personalized message. Example:

What is your name? John

How old are you? 15

Hello, John! You are 15 years old.

---

### Part 3: Conditional Logic

Add logic to tell the user if they are old enough to drive (assume the driving age is 18). Example:

```
You are too young to drive.  
-- or --  
You are old enough to drive.
```

---

### Part 4: Loops

Use a loop to print numbers from 1 to 5. Example:

```
1  
2  
3  
4  
5
```

**Bonus:** Counting is fundamental to many concepts. Use a loop to count down from 5 to 1. Another concept is to print only even or odd numbers.

---

### Part 5: Functions

Write a function called `greet_user(name)` that prints a greeting. Example:

```
Hello, Alice! Have a great day!
```

Call the function from your main program.

---

### Challenge (Optional): Number Guessing Game

- Generate a random number between 1 and 10.
- Ask the user to guess the number.
- Give feedback if the guess is too high, too low, or correct.
- Allow up to 3 guesses.

# Assignment 2

## Simple Calculator



Concepts Covered:

- Variables
- User input/output
- Conditionals
- Functions



**Objective:** Create a calculator that can perform basic operations: **addition**, **subtraction**, **multiplication**, and **division** between two numbers. Also, code should Include error-handling for division by zero.

### Sample Flow:

Enter first number: 12

Enter operator (+, -, \*, /): \*

Enter second number: 5

Result:  $12 * 5 = 60$

# Assignment 3

## Quiz Game



### Concepts Covered:

- Input/output
- Conditionals
- Functions
- Loops



**Objective:** Create a small 10-question quiz. The program should:

- Ask the user each question
- Check if the answer is correct
- Keep track of the score
- Show the final score at the end

### Example:

What is the capital of France? Paris

Correct!

...

Your score: 2/10

**Bonus:** Use a loop to let the user play again. Randomize order.

# Assignment 4

## Understanding Basic Memory Management

### Concepts Covered:

- Stack vs Heap memory
  - Variable scope and lifetime
  - Dynamic memory allocation and deallocation (malloc, free in C; new, delete in C++)
  - Memory leaks (conceptual or practical)
  - Pointers (for pointer-based languages)
- 

### Assignment Details

#### ◆ Part 1: Stack Memory (Automatic Allocation)

**Objective:** Write a function that declares local variables (integers, arrays) and prints their addresses (if language supports it).

Example in C:

```
void showStackMemory() {
    int x = 10;
    int arr[5];
    printf("Address of x: %p\n", (void*)&x);
    printf("Address of arr: %p\n", (void*)&arr);
}
```

---

#### ◆ Part 2: Heap Memory (Dynamic Allocation)

**Objective:** Write a program that dynamically allocates memory for:

- A single integer
- An array of n integers

Fill the array with values and print them. Then free the allocated memory.

Example in C:

```
int *arr = malloc(n * sizeof(int));
if (arr == NULL) {
    // handle error
}
...
free(arr);
```

---



### ◆ Part 3: Memory Leak Demonstration (Optional)

**Objective:** Deliberately create a memory leak by allocating memory and not freeing it. Use a tool (like **Valgrind**, if available) to detect the leak. Try a small memory leak because bigger memory leaks can crash programs and other undesired outcomes.

---

### ◆ Part 4: Pointer and Scope Test (C/C++ only)

**Objective** Write a function that returns the address of a local variable. Explain what happens and why it's problematic.

```
int* badFunction() {  
    int x = 5;  
    return &x; // Warning: returning address of a local variable  
}
```

---


### Challenge Task (Optional)

**Mini Project:** Create a simple student record system using dynamic memory allocation. Let the user:


- Allocate memory for n students
- Input name and ID
- Display the records by ID or by Alphabetical order by last name
- Search for student
- Free the memory at the end


# Project 1


## Touch-Activated LED Light

 **Objective:** Build a simple circuit that lights up an LED when you touch a capacitive sensor.

---

 **Component to Add:** Capacitive Touch Sensor

 **How It Works:** The LED lights up when you touch the sensor's surface, detecting a change in capacitance.

 **Real-Life Use:** Used in touch lamps, interactive exhibits, or as a button replacement in user-friendly interfaces.

---

### What You Need:

- Arduino Uno (or similar)
  - 1x LED
  - 220Ω resistor
  - Capacitive touch sensor (e.g., TTP223)
  - Jumper wires
  - Breadboard
- 

### How It Works:

When you touch the sensor, it sends a HIGH signal to one of the Arduino's digital pins. The Arduino then turns on the LED. When you let go, the LED turns off.

---

### Why It's Great for Beginners:

- Involves basic input and output
- Teaches pin configuration and logic control
- No complex components—easy to assemble and understand

# Project 2

## Extended Versions of the Buzzer Alarm Project

**Overall Objective:** The objective of this component project is to help students understand how code and hardware work together. Once a programmer grasps the fundamentals, a wide range of creative and practical possibilities become available.

### 1. Motion-Activated Alarm

**Component to Add:** PIR Motion Sensor

**How It Works:** The buzzer sounds when the sensor detects movement.

**Real-Life Use:** Basic home security system or intruder alert.

```
if (digitalRead(pirPin) == HIGH) {  
  tone(buzzerPin, 1000); // Start buzzer  
  delay(1000);  
  noTone(buzzerPin); // Stop buzzer  
}
```

---

### 2. Light-Triggered Alarm

**Component to Add:** Photoresistor (LDR)

**How It Works:** The buzzer turns on when the light level drops (e.g., when someone covers the sensor).

**Real-Life Use:** Anti-theft alert for a drawer or box.

```
int lightLevel = analogRead(ldrPin);  
if (lightLevel < threshold) {  
  digitalWrite(buzzerPin, HIGH);  
}
```

---

### 3. Delayed Alarm

**Feature to Add:** Countdown timer with LED indicator

**How It Works:** When a button is pressed (or motion is detected), it flashes an LED for 5 seconds before sounding the buzzer.

**Real-life Use:** Gives users time to cancel the alarm or leave the room.

---

#### 4. Password-Protected Alarm

**Component to Add:** Keypad or Serial Monitor

**How It Works:** The buzzer stays active until the correct code is entered.

---

#### Bonus Add-ons:

- OLED or LCD screen to display status messages
- Use `tone()` to play a melody when triggered
- Control the alarm remotely with an IR remote or Bluetooth

# Project 3

## Arduino Project: Student-Selected Component

### 1. Project Title

Provide a clear and concise title describing the purpose of the project.

*Example: "Temperature Logger Using DHT11 Sensor"*

---

### 2. Objective

State the primary goal of the project and what the student aims to achieve using the chosen component.

*Example: To read and display real-time temperature and humidity data using an LCD display.*

---

### 3. Component Overview

Briefly describe the selected component, including:

- What it is used for
  - Why it was chosen
  - Key features
  - Documentation Info
  - Image
- 

### 4. Required Materials

List all hardware and software used in the project:

- Arduino board model
  - Selected component
  - Additional components (e.g., resistors, breadboard, jumper wires)
  - Software (e.g., Arduino IDE, libraries)
-

## 5. Circuit Diagram

Provide a schematic or image showing how the components are connected to the Arduino. Since this is a programming class with an Arduino, we do not have to have strict documentation of this. However, we need to keep any reference material for submission. A hand-drawn or digital wiring diagram is acceptable.

---

## 6. Code

Include the complete source code. Ensure comments are used to explain key sections. Comment code on why code was completed this way. Code must be tested and should compile without errors.

---

## 7. Explanation of Code

Break down how the code works:

- Initialization
  - Main loop logic
  - How data is read, processed, or displayed
  - Any other relevant info
- 

## 8. Testing and Results

Document:

- How the system was tested
  - Observations or data collected
  - Any challenges encountered
- 

## 9. Conclusion

Summarize what was learned through the project:

- Did the component work as expected?
  - How could the project be improved?
  - Suggestions for future developments or extensions
-

## **10. References**

Include links or citations for:

- Datasheets
- Libraries used
- Tutorials or example projects referenced

# Resources

My YouTube channel features examples of working Arduino code in action. Each video corresponds to a specific folder in my GitHub repository, where you can find the full source code and related files. This setup allows you to follow along visually while accessing the exact code used in the demonstrations. If you have any questions or need further clarification, feel free to reach out to me directly. I am happy to help support your learning and project development.

There are many sources to get programming information from. The Arduino website is the official hub for everything related to Arduino, offering documentation, tutorials, and downloadable tools to help users get started with microcontrollers. Stack Overflow is a popular forum where programmers and makers can ask technical questions and receive answers from a global community of developers. It is especially helpful for troubleshooting errors and learning from real-world coding examples. W3Schools is a beginner-friendly site that teaches web development and programming basics through interactive lessons and live coding practice. Together, these resources provide a strong foundation for anyone learning to code or working on Arduino projects, from understanding syntax to solving problems and exploring hands-on applications. There are many choices to choose from when getting basic quality help from online sources.

YouTube: [https://www.youtube.com/channel/UCOv\\_iZCx4CW5Y9S8YzXDdgg](https://www.youtube.com/channel/UCOv_iZCx4CW5Y9S8YzXDdgg)

GitHub: <https://github.com/Vaughan-Peter?tab=repositories>

LinkedIn: <https://www.linkedin.com/in/peter-vaughan-997478239/>

E-mail: [otherhalifaxprojects@gmail.com](mailto:otherhalifaxprojects@gmail.com)

Arduino Website: <https://www.arduino.cc/>

Stack Overflow: <https://stackoverflow.com/questions>

W3 Schools: <https://www.w3schools.com/>