ARDUINO ASSIGNMENTS

Peter Vaughan

NSCC Online Learning

Contents

Arduino Coding Class

Textbook / Resource Requirements

No textbook is required.

Laptop or desktop with:

- Windows 10 or Later, MacOS or certain Linux distributions
- 1 GB of hard drive Space
- 1.6 GHz or faster processor
- 1 GB of RAM

Supplies / Additional Resources

1. Arduino Uno
2. USB 2.0 A Male to B Male Cord USB A to B Cable
3. 10 1k Ohm Resistors
4. 10 LEDs (any colors)
5. 30 Jumper Cables
6. Electronic Breadboard

Alternatives can be arranged if supplies are unavailable. The school should have some available for this purpose. Otherwise, we will look into a simulator.

Course Learning Outcomes

1) Learn basic coding (refer to the syllabus in the other documentation)
2) Basic code troubleshooting
3) Interfacing with hardware
4) Documentation

Evaluation Scheme

Assignments: 6 @ 10% each	60%
Project: 1 @ 30% each	30%
Self-Study: 1 @ 10% each	10%
Total	**100%**

Assignment #1

C++ Environment Setup and Basic Programming Practice

Objective: Set up your C++ development environment using Visual Studio Code (VS Code), and write simple programs that demonstrate understanding of basic programming concepts.

Part 1: Install & Configure VS Code for C++

Instructions:

1. Download and install VS Code.
2. Install the C++ extension for VS Code (C/C++ by Microsoft).
3. Install a C++ compiler (GCC via MinGW for Windows, or Clang for macOS/Linux).
4. Create and run a "Hello, World!" C++ program to confirm everything is working.

Submission Requirement: Screenshot of VS Code with your "Hello, World!" program running in the terminal.

Part 2: C++ Programming Practice

Write **three separate C++ programs**, each demonstrating one of the following programming concepts:

1. Variables: Write a program that declares and initializes different types of variables (int, float, string). Display their values using `cout`.

2. Conditionals: Write a program that asks the user for their age and uses an `if-else` statement to output:

- "You are a minor" if age < 18
- "You are an adult" if age ≥ 18

3. Loops: Write a program that uses a `for` loop to print numbers from 1 to 10. Add a simple `while` loop that does the same, for extra practice.

Optional Reflection (Bonus: +1 mark): In a short paragraph, describe your experience installing the tools and completing the exercises. What did you find easy or difficult?

Assignment #1 Rubric

This assignment is worth 10% of the final grade. Students are required to install the VS Code IDE for C++, write three simple C++ programs demonstrating variables, conditionals, and loops, and submit their work to GitHub. Code is the most heavily weighted component.

Criteria	Description	Marks
GitHub Repo of All Code	Repository includes complete, working code for variables, conditionals, and loops with clear filenames and a README.	5 marks
Brief Summary	Includes a reflection on what you did, what you liked, and what you struggled with.	2.5 marks
Photos	Includes screenshots showing VS Code installed and code execution output.	2.5 marks

Submissions

Required Submission for marks:

- GitHub Repo Link of all code
- Brief summary of what you did, what you liked and what you struggled with
- Photos of installed IDE and other working code

Assignment #2

Basic C++ Coding Applications

Objective: Assignment #2 has three parts to grow on our foundation of programming. Students will design their own calculator with input, output, and error handling. Then students will create an "Even/Odd Checker" with looping and conditionals. Students will create a their own "Factorial Calculator". Students can be creative on how to approach these programs. These three basic C++ codes will together cover the fundamentals of:

- Variables
- Conditionals
- Loops
- Functions
- Basic Input/Output
- Error Handling

1. Calculator with Input, Output, and Error Handling

Create your own calculator. It does not need to be complicated or require a GUI. It is recommended to learn with the IDE's Terminal. This will be shown in class.

Concepts Covered:

- Variables
- Input/Output
- Conditionals
- Error handling (e.g., division by zero)

2. Looping and Conditionals: Even/Odd Checker

Create your own Even/Odd Checker. It does not need to be complicated or require a GUI. It is recommended to learn with the IDE's Terminal. This will be shown in class.

Concepts Covered:

- Loops
- Conditionals
- Input/Output

3. Functions: Factorial Calculator

Create your own Even/Odd Checker. It does not need to be complicated or require a GUI. It is recommended to learn with the IDE's Terminal. This will be shown in class.

Concepts Covered:

- Functions
- Loops
- Input/Output
- Basic error checking

Assignment #2 Rubric

The complete assignment is worth 10% of the final grade. Students are required to install the VS Code IDE for C++, write three simple C++ programs demonstrating variables, conditionals, and loops, and submit their work to GitHub. Code is the most heavily weighted component.

Criteria	Description	Marks
GitHub Repo of All Code	Repository includes complete, working code for variables, conditionals, and loops with clear filenames and a README.	5 marks
Brief Summary	Includes a reflection on what you did, what you liked, and what you struggled with.	2.5 marks
Photos	Includes screenshots showing VS Code installed and code execution output.	2.5 marks

Submissions

Required Submission for marks:

- GitHub Repo Link of all code
- Brief summary of what you did, what you liked and what you struggled with
- Photos of installed IDE and other working code

Assignment #3

Assignment: Build a Mini Student Grading System

Objective: Write a C++ program that takes input for multiple students, calculates their grades, and handles invalid inputs gracefully. Small sample code should be provided during class time. The program must be modular using functions and demonstrate all core programming basics, which are:

- Variables
- Conditionals
- Loops
- Functions
- Basic input and output
- Error Handling
- Some documentation

Documentation & Reflection

Include a short document (PDF or Word file) with:

- **Program Summary:** What your program does and how it works
- **Issues:** Summary of any issues you had
- **Screenshots:** Showing program input/output and error handling
- **Reflection:** What you learned, what was easy/hard

Assignment #3 Rubric

Assignment 3 is worth 10% of the final grade. Each part of the submission is equally weighted, and students must complete all components to receive full marks.

Criteria	Description	Marks
GitHub Repository	Complete and organized project code with commit history and README file.	2.5 marks
Documentation	Clear explanation of program functionality and instructions for running/testing.	2.5 marks
Brief Summary	Reflection on what was done, what you liked, and challenges encountered.	2.5 marks
Photos of Working Program	Clear images showing the program running or hardware setup.	2.5 marks

Submissions

Required Submission for marks:

- GitHub Repo Link
- Documentation
- Brief summary of what you did, what you liked and what you struggled with
- Photos of installed IDE and other working code

Assignment #4

Interactive Quiz Program

Objective: Students create a program that asks multiple-choice questions, accepts user input, checks answers, and gives feedback. This task covers essential programming logic fundamentals. Sample code will be giving during class time to help students get started.

Programming Concepts Covered:

- **Variables:** Storing questions, answers, and scores
- **Input/Output:** Getting user input and displaying messages
- **Conditionals:** Using if, else if, and else to check answers
- **Loops:** Repeating questions or allowing multiple attempts
- **Functions:** Organizing code into reusable blocks (optional, for advanced learners)
- **Data Structures:** Arrays or lists to store questions and answers

Example Outline:

1. Display a question with multiple choices
2. Read user input (e.g., 1, 2, 3, or 4)
3. Check if the answer is correct using conditional statements
4. Keep track of the score
5. At the end, display the total score

Assignment #4 Rubric

Assignment 4 is worth 10% of the final grade. Each part of the submission is equally weighted, and students must complete all components to receive full marks. Refer to my GitHub for examples on how to do these outcomes. See “Additional Info” at the end this document for my GitHub link.

Criteria	Description	Marks
GitHub Repository	Complete and organized project code with commit history and README file.	2.5 marks
Documentation	Clear explanation of program functionality and instructions for running/testing.	2.5 marks
Brief Summary	Reflection on what was done, what you liked, and challenges encountered.	2.5 marks
Photos of Working Program	Clear images showing the program running or hardware setup.	2.5 marks

Submissions

Required Submission for marks:

- GitHub Repo Link
- Documentation
- Brief summary of what you did, what you liked and what you struggled with
- Photos of working code

Assignment #5

Assignment: Arduino IDE Setup and Basic Programming

This assignment is designed to ensure students can successfully install the Arduino IDE, write a simple Arduino sketch, and document the process. The goal is to have student be familiar with the Arduino environment and demonstrate basic microcontroller programming skills.

Objectives: Install the Arduino IDE on your computer. Write and upload a basic sketch to an Arduino board (e.g., Blink). Document your setup and code with screenshots and explanations.

Assignment #5 Rubric

This assignment is worth 10% of the final grade. Students are required complete all aspects of the assignment. Code is the most heavily weighted component.

Criteria	Description	Marks
GitHub Repository	Includes working Arduino sketch with clear commit history and README.	4 marks
Documentation	Describes installation, coding process, and includes reflection.	3 marks
Pictures	Screenshots of IDE installation and running project (e.g., blinking LED).	3 marks

Submissions

Required Submission for marks:

- GitHub Repo Link
- Documentation
- Pictures of IDE and working assignment
- Brief summary of what you did, what you liked and what you struggled with

Assignment #6

Assignment: 9-LED Arduino Project with Basic Programming Concepts

This assignment will help you develop and demonstrate essential Arduino programming skills using a simple 9-LED setup. Students will install the Arduino IDE, write and test a sketch that controls 9 LEDs, and document your work. Your code should demonstrate variables, conditionals, loops, functions, input/output, error handling, and basic serial communication.

Objectives:
- Install and use the Arduino IDE.
- Connect and control 9 LEDs using an Arduino board.
- Demonstrate basic programming principles in your code.
- Document the development and testing process.

Assignment #6 Rubric

A complete 9-LED Arduino project demonstrating core programming logic, serial communication, error handling, and supported by well-documented code and GitHub submission.

Criteria	Description	Marks
Code Functionality	Controls 9 LEDs using appropriate logic (variables, loops, conditionals, functions).	4 marks
Serial Communication	Demonstrates basic serial communication for debugging or interaction.	1 mark
Error Handling	Handles possible input or hardware issues gracefully in code.	1 mark
GitHub Repository	Complete and clearly organized with README and commit history.	2 marks
Documentation	Explains the build and thought process with visuals and reflection.	2 marks

Submissions

1. GitHub Repository Link containing:
   - Your complete Arduino sketch (e.g., `nine_leds.ino`)
   - A brief README explaining your code
2. Documentation (PDF or Word format) including:
   - What you did and why
   - What you liked and struggled with
   - Screenshots of your IDE, wiring setup, and working project

Project #1

Students may look through my GitHub to get started on a project. Students may select from one of the project ideas:

- Blinking LEDs
- LEDs Fading
- Traffic Light Simulation
- LED Chaser
- Binary Counter
- LED Dice
- Morse Code Flasher
- Heartbeat Simulator (Pulse LED)
- A communication concept between Arduino and PC

Otherwise, students may submit a proposal idea for approval. Include what your project is and what you expect to learn. Once approved, the student can work on that project for a submission.

Arduino LED Project Rubric

This rubric is used to evaluate student submissions for a simple Arduino LED project. The total score is out of 30 marks, with the highest weight given to documentation.

Criteria	Description	Marks
GitHub Repository	Project is uploaded to GitHub. Includes code, organized structure, and README.	5 marks
Documentation	Clear instructions, wiring diagrams (if needed), code comments, and explanations. Covers how it works, and how to recreate it.	12 marks
Pictures / Visual Evidence	At least 2 clear pictures or a short video of the actual project.	5 marks
Summary & Reflection	Short paragraph on what was done, what was enjoyable, and what challenges were faced.	8 marks

Submissions

Required Submission for marks:

- GitHub Repo Link
- Documentation
- Pictures or a video of a working project
- Brief summary of what you did, what you liked and what you struggled with

Self-Study Project

You can explore any mini-Arduino project. You may choose one idea from the project list, only if it was not previously done. Otherwise, submit a project idea for approval. Include what your project is and what you expect to learn. Once approved, the student can work on that project for a submission.

Arduino Self-Study Rubric

This rubric is used to evaluate student submissions for a simple Arduino self-study project. The total score is out of 10 marks, with the highest weight given to the code in the GitHub repository.

Criteria	Description	Marks
GitHub Repository (Code)	Code is uploaded, working, and well-structured. Includes README and file organization.	4 marks
Documentation	Clear explanation of how it works, includes serial communication flow or wiring if applicable.	2 marks
Pictures / Evidence	At least two photos or screenshot of the serial monitor or setup.	2 marks
Summary & Reflection	Short paragraph describing what you did, what you liked, and challenges faced.	2 marks

Submissions

Required Submission for marks:

- GitHub Repo Link
- Documentation
- Pictures
- Brief summary of what you did, what you liked and what you struggled with

Overall Comments

Please refer to the document in my GitHub repo titled “Public General Use - Mockup of Class Time” for a proposed class schedule and potential topics. There is a vast range of concepts that can be taught using Arduino, and these naturally extend to platforms like Raspberry Pi and embedded systems as a whole. The programming and hardware skills developed through these tools are highly transferable—across different programming languages, platforms, and design methodologies.

Additional Info

GitHub Repo: https://github.com/Vaughan-Peter/ArduinoLearning

Contact Info: https://www.linkedin.com/in/peter-vaughan-997478239/

Contact E-mail: otherhalifaxprojects@gmail.com

Bonus Activities

Number Guessing Game

Description: Write a program where the computer randomly selects a number between 1 and 100, and the user tries to guess it. After each guess, the program tells the user if the guess was too high, too low, or correct.

Why it's fun and effective:

- Uses **variables**, **conditionals**, **loops**, and **input/output**
- Offers instant feedback and a bit of challenge
- Easy to expand (e.g., track number of attempts, add difficulty levels)

LED Chase Game

Goal: Make the LEDs "chase" each other in sequence. Then, add a twist: randomly reverse the direction or change speed using basic logic.

Concepts Covered:

- **Variables –** Track LED state and direction
- **Loops –** Create the chase effect
- **Conditionals** – Change direction/speed logic
- **Functions** – Organize behavior (e.g., `chaseForward()`, `chaseReverse()`)
- **Basic I/O** – Control LEDs with `digitalWrite()`
- **(Optional)**: Use `Serial.print()` for debugging the direction or state

Setup:

- Use 5–9 LEDs
- Connect each LED to a digital pin through a resistor
- Make sure all LEDs share a common ground

Challenges to Add:

- Change chase direction every few seconds
- Speed up over time
- Make the middle LED "pulse" when idle