Fundamentals of Software Development for Electronics

Prepared By: Dr. Rony Ibrahim

Course outline

- Session 1: Importance of Software Engineering in Electronics
- Session 2: Basic Concepts and Terminologies
- Session 3: Software Development Life Cycle (SDLC)
- Session 4: Integrated Development Environments (IDEs) and Basic Git Workflow
- Session 5 and 6: JavaScript & Introduction to TypeScript
- Session 7 and 8: Building Web Applications with ReactJS
- Session 9: CSS Basics and Responsive Design Principles
- Session 10: Basics of Flutter / React Native / Ionic
- Session 11: Project Presentation & Final Review

Integrated Development Environments (IDEs) and Basic Git Workflow

Objectives

- Understand how to set up and configure Integrated Development Environments (IDEs) for software development in electronics projects.
- Learn the basics of Git and its workflow for version control and collaboration.
- Explore advanced Git features for collaborative projects in embedded systems.

What is an IDE?

Definition: An Integrated Development Environment (IDE) is a software application that provides comprehensive tools to facilitate software development.

- **Code Editor:** Write and manage code files.
- **Compiler/Interpreter:** Compile or interpret code into machine-executable instructions.
- **Debugger:** Find and fix errors in code.
- **Terminal/Command Line:** Run scripts and commands directly from the IDE.

Popular IDEs for Electronics:

- **VSCode**, PyCharm, Eclipse, PlatformIO (for embedded systems).

IDE Setup for Embedded Systems Development

- VSCode with Extensions:
  Install extensions for Js, react, Python, C/C++, and embedded systems development (PlatformIO, Arduino).
- PyCharm:
  Ideal for Python-based projects on platforms like Raspberry Pi.
- PlatformIO:
  Multi-platform IDE specifically for embedded systems.
  Supports frameworks like Arduino, etc.
- Eclipse
  Widely used in industrial embedded system projects, supports C/C++.

IDE Setup for Raspberry Pi Projects

Setting up Visual Studio Code (VSCode):

- Install Node.js and TypeScript for JS/TS development.
- Install Remote - SSH plugin for Raspberry Pi access from VSCode.
- Configure the environment to run code directly on the Raspberry Pi.

Remote Development: How VSCode can be used to write code on your local machine but execute it on a Raspberry Pi.

Git and Version Control Basics

What is Git?
A distributed version control system that tracks changes in code and manages collaborative development.

Why Git in Electronics Projects?

- Tracks changes to firmware, scripts, and software configurations.
- Enables collaboration among multiple developers.

Basic Workflow:

- Create or **Clone** a project / Makes Changes (features or bug fixing) / **Commit** / **Pull** / **Push** / **Pull Requests** and Reviews

Git and Version Control Basics

- Repository: A project folder tracked by Git.
- Commit: A snapshot of changes in the project.
- Branch: A parallel version of your project for working on new features.
- Merge: Integrating changes from one branch into another.

Why Git is important in team-based and long-term projects like your Raspberry Pi project?

Basic Git Commands

Essential Commands: (Use Vscode extension Gitlens)

- ***git clone [repo URL]***: Clone a remote repository.
- ***git status***: Check the current status of the local repository.
- ***git add [file]***: Stage changes for commit.
- ***git commit -m "message"***: Commit changes with a message.
- ***git push***: Push changes to the remote repository.
- ***git pull***: Fetch and integrate changes from the remote repository.
- git fetch

Git Branches and Collaboration

What are Branches?

Isolated environments within a project where developers can work on new features or fixes without affecting the main codebase.

Creating Branches:

- ***git branch [branch-name]***: Create a new branch.
- ***git checkout [branch-name]***: Switch to the branch.

Merging Branches:

- Once a feature or fix is complete, branches are merged back into the main branch (typically *master* or *main*).
- ***git merge [branch-name]***: Merge changes from another branch.

Exercise 1 – Set Up a Project in VSCode for Raspberry Pi

Task: Set up a simple JS/TS project in VSCode that connects to your Raspberry Pi using the Remote - SSH extension.

Instructions:

- Install the Remote - SSH extension in VSCode.
- Connect to your Raspberry Pi through SSH.
- Write a simple JavaScript/TypeScript program that turns an LED on or off using the Raspberry Pi's GPIO pins.
- Commit and push the project to GitHub.

Hint: Use the ***onoff*** library to control the Raspberry Pi GPIO pins from Node.js.

Exercise 1 – Solution script

```
const Gpio = require('onoff').Gpio;

// Set up the GPIO pin 17 as an output
const LED = new Gpio(17, 'out');

// Turn the LED on
LED.writeSync(1);

// Turn the LED off after 5 seconds
setTimeout(() => {
    LED.writeSync(0);
    LED.unexport();
}, 5000);
```

Exercise 2 – Using Git for Collaboration

Task: Practice the basic Git workflow by collaborating with a partner on a Raspberry Pi project.

Instructions:

- Clone a GitHub repository where the project will be hosted.
- Create a new branch, e.g., add-feature-x.
- Write a function in TypeScript to control multiple GPIO pins (e.g., controlling multiple LEDs or motors).
- Push the changes back to the repository and create a pull request.
- Review your partner's pull request and merge the changes.

Challenge: Add a feature that takes input from a sensor (e.g., temperature sensor) and controls the output based on the input.

Exercise 2 – Solution

```
import { Gpio } from 'onoff';

// Set up multiple GPIO pins
const LED1 = new Gpio(17, 'out');
const LED2 = new Gpio(18, 'out');

// Function to turn LEDs on and off
function controlLEDs(state1: number, state2: number) {
    LED1.writeSync(state1);
    LED2.writeSync(state2);
}

// Example: Turn both LEDs on
controlLEDs(1, 1);

// Turn them off after 5 seconds
setTimeout(() => {
    controlLEDs(0, 0);
    LED1.unexport();
    LED2.unexport();
}, 5000);
```

Project Challenge - Raspberry Pi Project Setup

Task: Prepare the foundation for your final Raspberry Pi project using Git and VSCode.

Instructions:

- Set up a new project on GitHub for your Raspberry Pi final project.
- Create a README file with a description of the project and the initial setup instructions.
- Write a basic script (in JS/TS) that demonstrates simple GPIO pin control.
- Push this to the repository and invite a teammate to collaborate on the project.
- Ensure that all commits are well documented with meaningful messages.

Objective: Lay the groundwork for the final project, integrating Git and VSCode for collaborative development.

Recap & Key Takeaways

Key Takeaways::

- IDEs like VSCode provide powerful tools for software-electronics development.
- Git is essential for managing code and collaborating on projects.
- Using VSCode's SSH feature, you can remotely develop software on a Raspberry Pi.
- Mastery of these tools will be crucial for success in the final Raspberry Pi project.