Internet Of Things

LAB Record

Course Code : BCS-508

|  |
| --- |
| Name: Abhinash Ray  Registration Number: FOS/BCS/2021-24/011  Course: B.Sc. Computer Science  Semester: 5th Semester  Faculty Teacher: Mr. Biswajeeban Mishra |

|  |  |
| --- | --- |
| Remarks |  |
| Signature |  |



INDEX

|  |  |  |  |
| --- | --- | --- | --- |
| Sl.  No. | NAME OF EXPERIMENTS | PAGE  NO. | REMARK |
| 1. | Basics of Git | 1-7 |  |
| 2. | Installing and Running Mosquitto MQTT | 8-9 |  |
| 3. | Controlling an LED with Arduino | 10-11 |  |
| 4. | Interfacing with DHT Sensor | 12-13 |  |
| 5. | Interfacing with Ultrasonic Sensor | 14-16 |  |

**EXPERIMENT 1: GIT**

**Basics:**

Q) What is version control? Why is it important?

Ans) Version control is a system that enables developers and teams to track and manage changes to their code, documents, and other digital assets over time. It provides a structured way to organize, document, and collaborate on projects by keeping a history of all changes made to files and directories. Version control systems (VCS) help developers work together efficiently and ensure the integrity and stability of their codebase.

**Version control is important because:**

**History and Tracking**: Version control systems maintain a complete history of every change made to a project. This means you can see who made a change, when it was made, and what exactly was changed.

**Backup and Recovery**: Version control acts as a reliable backup system. If files are accidentally deleted or corrupted, you can easily restore previous versions.

**Conflict Resolution**: When multiple developers modify the same code simultaneously, version control systems provide tools to help resolve conflicts.

Q) What's the difference between Git and GitHub?

Ans) **Git is a Version Control System (VCS):** Git is a distributed version control system that allows developers to track and manage changes to their source code or other text-based files.

Git is the version control system itself, which you install and use on your local machine.

**GitHub is a Hosting Service for Git Repositories:** GitHub is a web-based platform that provides hosting for Git repositories.

GitHub is a popular choice for hosting, sharing, and collaborating on Git repositories, especially in the open-source community and among development teams.

Q) Describe the Git workflow (add, commit, push, pull).

Ans) The Git workflow involves a series of commands and actions that developers use to track changes to their code, collaborate with others, and maintain a version-controlled codebase.

**Add (git add):**

Before Git records changes in your project, you need to specify which files or changes you want to include in the next commit. This is done using the git add command.

**Commit (git commit):**

The git commit command saves the staged changes to the Git repository and moves the project's history forward.

**Push (git push):**

The git push command is used to upload your local commits to a remote repository. This makes your changes accessible to others who have access to the same remote repository.

**Pull (git pull):**

The git pull command is used to fetch changes from a remote repository and merge them into your local branch automatically. It combines the git fetch (to retrieve changes) and git merge (to integrate changes) steps into a single command.

Q) What is a repository in the context of Git?

Ans) In the context of Git, a repository (also known as "repo") refers to a data structure that stores all the files, history, and metadata for a specific project. It is a fundamental concept in version control and is used to manage and track changes to a set of files and directories over time.

**Commits:**

Q) What is a commit in Git?

Ans) a commit is a concept that represents a snapshot of the entire codebase at a specific point in time. It's a record of the changes made to the files and directories in a Git repository. Each commit contains the following information:

1.Author Information.

2.Timestamp.

3.Commit Message.

4.Changes.

Q) How is each commit uniquely identified?

Ans) Each commit in Git is uniquely identified by a hash, which is a long string of characters. This hash is generated based on the contents of the commit and its parent commit(s), ensuring that no two commits will have the same identifier.

**Remote Repositories:**

Q) What is a remote repository in the context of Git?

Ans) In the context of Git, a remote repository (often simply referred to as "remote") is a repository that exists on a different location or server from your local Git repository. A remote repository in Git serves as a central hub for sharing, collaborating, and synchronizing code changes among multiple developers or teams.

Q) What are the default names that Git uses for the repository you cloned from and your local repository?

Ans) If we clone a repository, the command automatically adds that remote repository under the name “origin”. So, git fetch origin fetches any new work that has been pushed to that server since you cloned it.

Q)How do you synchronize changes from a remote repository to your local one, and vice versa?

Ans) **To Synchronize Changes from Remote to Local (Fetch and Merge)**

**Step 1: Fetch Remote Changes (**git fetch**)**

**Step 2: Merge Remote Changes into Your Local Branch (**git merge **or** git pull**)**

**To Synchronize Changes from Local to Remote (Push)**

**Step 1: Commit Your Local Changes (**git commit**)**

**Step 2: Push Local Commits to the Remote Repository (**git push**)**:

**GitHub Specifics:**

Q) What is a pull request?

Ans) A pull request is a feature in Git-based version control systems that allows developers to propose changes they've made in their own fork or branch of a repository to be merged into the original repository. It serves as a formal way to request code review, collaboration, and integration of changes into the main codebase. Pull requests are commonly used in collaborative software development to maintain code quality and facilitate team coordination.

Q)How do you 'fork' a repository on GitHub, and why might you want to?

Ans) To 'fork' a repository on GitHub, you can click the "Fork" button on the top right of the repository's page. Forking creates a personal copy of the repository under your GitHub account. You might want to fork a repository to:

**Contribute:** Make changes to someone else's project.

**Experiment:** Test changes or explore new ideas without affecting the original.

**Collaborate:** Collaborate on a project with others, using forks as separate workspaces.

**Create a backup:** Safeguard a snapshot of a project for future reference.

Q) How can you use GitHub to collaborate on open-source projects?

Ans) To collaborate on open-source projects using GitHub:

**Fork the Repository:** Fork the project's repository to create your own copy.

**Clone Your Fork:** Clone your forked repository to your local machine.

**Create a Branch:** Create a new branch for your work.

**Make Changes:** Make and commit your changes locally.

**Push Changes:** Push your branch with changes to your fork on GitHub.

**Create a Pull Request:** Open a pull request to propose your changes to the original repository.

**Discuss and Review:** Collaborate with project maintainers and contributors through

**Collaboration and Best Practices:**

Q) Why is it important to write clear commit messages?

Ans) Clear commit messages are important because they:

Enhance Readability.

Aid Debugging.

Document History.

Improve Maintainability.

Q) When collaborating with others, why might it be important to frequently pull the latest changes?

Ans) Frequently pulling the latest changes is important when collaborating because it helps:

**Stay Synchronized:** Ensure your local copy is up to date with others' contributions.

**Prevent Conflicts**: Reduce the likelihood of code conflicts when multiple people are making changes.

**Facilitate Integration:** Simplify the process of merging your changes into the latest codebase.

**Enhance Collaboration:** Promote real-time collaboration and coordination among team members.

**EXPERIMENT 2: Mosquitto MQTT**

**Installation of Mosquitto MQTT**

Step 1: Download the 64-bit or 32-bit installer from mosquito.org based on your device specifications

Step 2: Open the installer. Click next to begin installation

Step 3: Select the components you wish to install

Step 4: Select the directory to which you want to install the files

Step 5: After completion, click finish

**Adding Mosquitto to system path:**

Step 1: Open Advanced System Settings in properties of the File Explorer

Step 2: Click on Environment Variables

Step 3: Under system variables, locate the variable “Path” and click edit

Step 4: In the Edit Environment Window, click new and browse to the mosquitto installation folder and click OK.

**Publisher-Subscriber Model**

The pub-sub model is a messaging pattern where systems communicate through a central hub. The subscribers select multiple topics about which they receive messages. The publisher generates these messages and sends all the various messages of all the different topics to the respective subscribers.

**Verifying Mosquitto Installation**

To verify that Mosquitto has been added to your system path, open a new command prompt window (old command prompt windows won’t reflect the change) and type mosquitto -v. This should print the version of Mosquitto to the console if Mosquitto was added to the system path successfully.

By following these steps, you should be able to invoke Mosquitto from any location within the command line on your Windows system.



**EXPERIMENT 3: Controlling an LED with Arduino**

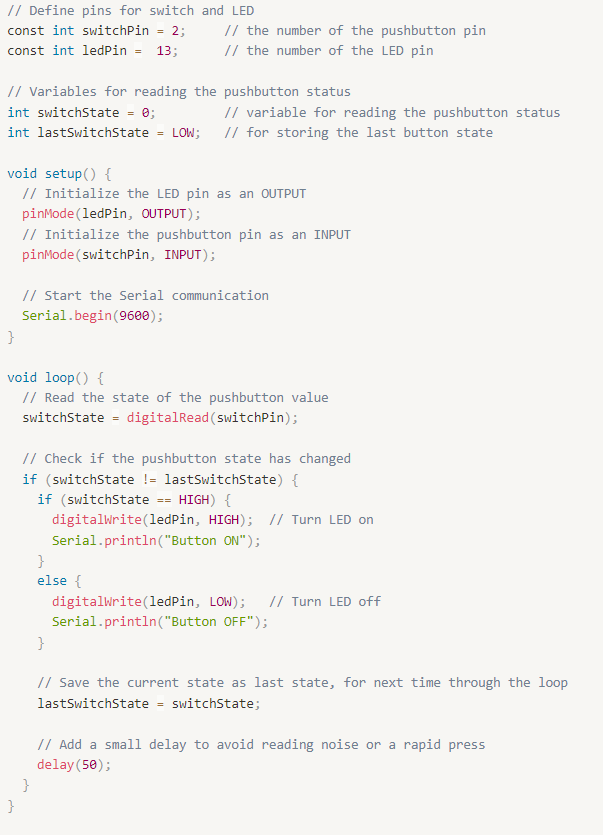
**Hardware Setup:**

* Connect one terminal of the switch to digital pin 2 (or any other digital pin) on the Arduino.
* Connect the other terminal of the switch to Ground (GND) through a 10kΩ resistor (this is a pull-down resistor).
* Connect the anode (longer leg) of the LED to digital pin 13 (or any other digital pin) on the Arduino via a 220Ω resistor.
* Connect the cathode (shorter leg) of the LED directly to Ground (GND).

**Arduino Code:**

****

**Code to display the ON/OFF Button:**

****

**EXPERIMENT 4: Interfacing with DHT Sensor**

**Hardware Setup:**

* Connect the DHT sensor's VCC to 5V, GND to Ground, and DATA pin to a digital pin, let's say pin 7.
* Set up the button as previously explained

**DHT Library:**

* Go to Arduino IDE > Sketch > Include Library > Manage Libraries.
* Search for "DHT sensor library" and install it

**Arduino Code:**

****

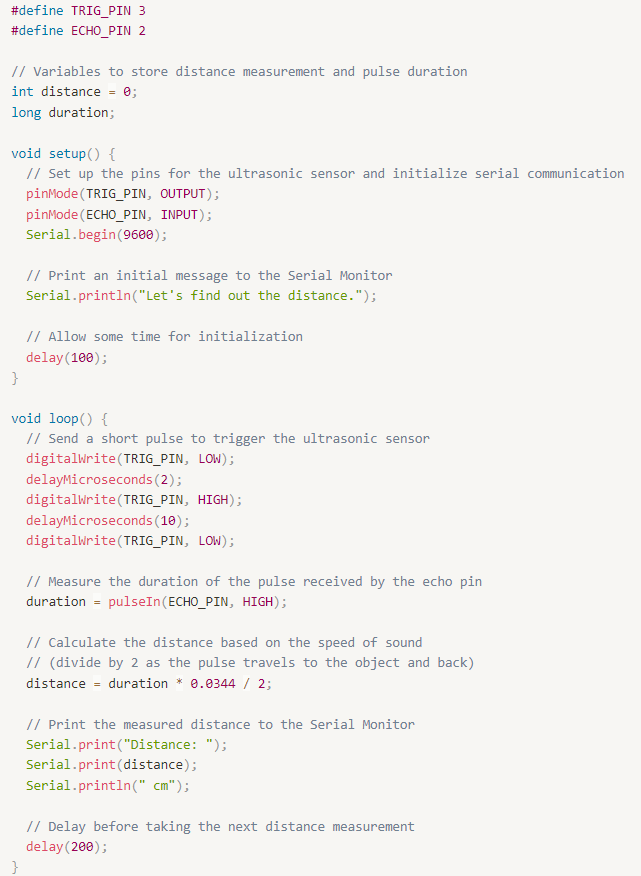
**EXPERIMENT 5: Interfacing with Ultrasonic Sensor**

**Hardware Setup:**

1. **Connect VCC and GND:**
   * Connect the VCC (power) pin of the ultrasonic sensor to a 5V pin on your Arduino.
   * Connect the GND (ground) pin of the ultrasonic sensor to a GND pin on your Arduino.
2. **Connect Trigger and Echo Pins:**
   * Connect the TRIG (trigger) pin of the ultrasonic sensor to digital pin 3 on your Arduino.
   * Connect the ECHO (echo) pin of the ultrasonic sensor to digital pin 2 on your Arduino.
3. **Serial Communication:**
   * If you plan to monitor the results on your computer, connect the Arduino to your computer using a USB cable.
4. **Upload Code:**
   * Open the Arduino IDE on your computer.
   * Copy and paste the provided Arduino code into the IDE.
   * Select your Arduino board type and port in the Arduino IDE.
   * Click the "Upload" button to upload the code to your Arduino.
5. **Open Serial Monitor:**
   * After uploading the code, open the Serial Monitor in the Arduino IDE.
   * Set the baud rate to 9600 (matching the **Serial.begin(9600);** in the code).
6. **Observe Results:**
   * Once the code is uploaded and the Serial Monitor is open, you should see messages indicating the distance measurements.
   * The distance in centimeters will be displayed, updating every 200 milliseconds.
7. **Verify and Troubleshoot:**
   * Ensure that the wiring is correct and secure.
   * If the sensor is not working as expected, check the connections, and make sure there are no loose wires.
   * Adjust the position of the ultrasonic sensor and the object being measured.
8. **Power Off Safely:**
   * When done experimenting, disconnect the power to your Arduino.

By following these steps, you should be able to set up the ultrasonic sensor with your Arduino and observe distance measurements in the Serial Monitor

**Arduino Code:**

****