**IoT LAB Record**



Course Code: BCS-508

|  |  |
| --- | --- |
| Name | Prateek Pandey |
| Registration Number | FOS/BDS/21-24/010 |
| Program | B. Sc (H.) Data Science |
| Semester | 5th |
| 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 Mosquito 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: Basics of GIT**

What is version control? Why is it important?

Version control is a system that tracks changes in files and their history over time.

It's important for collaboration, history tracking, and the ability to revert to previous states.

What's the difference between Git and GitHub?

Git

A distributed version control system for tracking changes in files on your local machine.

GitHub

A web-based platform for hosting Git repositories, facilitating collaboration, and offering additional features like pull requests and issue tracking.

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

add

Stage changes for commit.

commit

Record changes with a message.

push

Upload local changes to a remote repository.

pull

Fetch and merge changes from a remote repository.

What is a repository in the context of Git?

A repository is a directory that contains project files, their history, branches, and configuration. It's where Git stores and manages project versions.

Commits:

What is a commit in Git?

A commit in Git is a snapshot of your repository at a specific point in time.

It records changes made to files and includes a message describing the changes.

How is each commit uniquely identified?

Each commit is uniquely identified by a hash (a long alphanumeric string) generated based on the commit's content.

Remote Repositories:

What is a remote repository in the context of Git?

A remote repository in Git is a copy of your project hosted on a remote server.

It's separate from your local repository and allows for collaboration and sharing code with others. Common remote repository hosting services include GitHub, GitLab, and Bitbucket.

Default names that Git uses for the repository you cloned from and your local repository?

The default name Git uses for the repository you cloned from is typically called origin.

It refers to the remote repository from which you cloned your project.

The default name for your local repository is the name of your project directory.

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

To synchronize changes from a remote repository to your local one, use git pull.

This fetches changes from the remote and merges them into your local branch.

To synchronize changes from your local repository to a remote one, use git push.

This uploads your local changes to the remote repository, making them accessible to others.

GitHub Specifics:

What is a pull request?

A pull request (PR) on GitHub is a way to propose changes from one branch (or fork) of a repository to another. It allows collaborators to review, discuss, and potentially merge the changes. PRs facilitate code review and collaboration in a controlled and organized manner.

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

To fork a repository on GitHub, you click the "Fork" button on the repository's GitHub page.

This action creates a personal copy (fork) of the original repository under your GitHub account.

You might want to fork a repository to:

Contribute to open-source projects without direct write access.

Experiment with changes without affecting the original project.

Create your own version of a project for personal use or development.

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

Create GitHub Account

Sign up for a GitHub account if you don't have one.

Fork the Repository

On the project's GitHub page, click "Fork" to create a copy under your account.

Clone Your Fork

Use git clone to get a local copy of your forked repository.

Create a New Branch

Isolate your work with a new branch using `git checkout -b branch-name`.

Make Changes

Modify the code locally.

Commit Changes

Use git add and git commit to record changes.

Push Changes to Fork

Upload changes with git push origin branch-name.

Create a Pull Request

On GitHub, open a PR, specifying base and compare branches.

Collaborate

Reviewers provide feedback, discuss, and suggest improvements.

Merge PR

Once approved, a maintainer merges your changes into the main project.

Keep Fork Updated

Periodically sync your fork with the main project.

Continue Contributing

Contribute to more projects and collaborate with the community.

Collaboration and Best Practices:

Why is it important to write clear commit messages?

Clear commit messages are important because they:

1. Enhance Communication

Help team members understand changes.

1. Aid Documentation

Create a historical record of code changes.

1. Streamline Collaboration

Facilitate code review and issue tracking.

1. Improve Maintenance

Assist in identifying and fixing issues.

1. Ensure Team Alignment

Promote consistency and clarity.

1. Boost Code Review Efficiency

Enable quicker assessment of changes.

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

Stay Updated

Ensure you have the most current code, including bug fixes and new features.

Prevent Conflicts

Reduce the chances of merge conflicts by incorporating others' changes early.

Maintain Compatibility

Ensure your code works with the latest version of the project.

Collaborate Smoothly

Facilitate seamless teamwork by working on the same codebase.

Receive Feedback

Get timely feedback on your own changes and provide feedback on others'.

Avoid Divergence

Prevent codebase divergence that can lead to integration challenges later.

**EXPERIMENT 2: Mosquito MQTT**

**Installation of Mosquito 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 Mosquito 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 mosquito 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 Mosquito Installation**

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

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



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

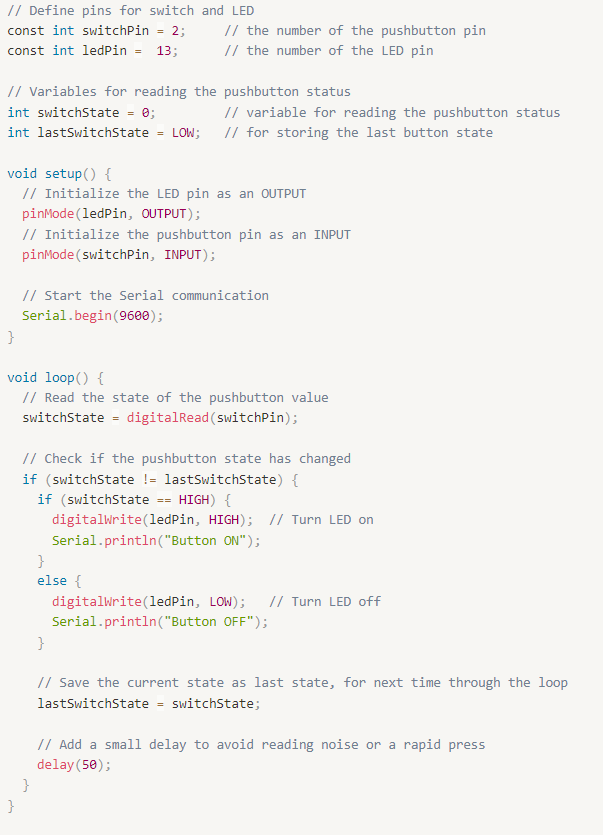
**Hardware Setup:**

* 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:**

****

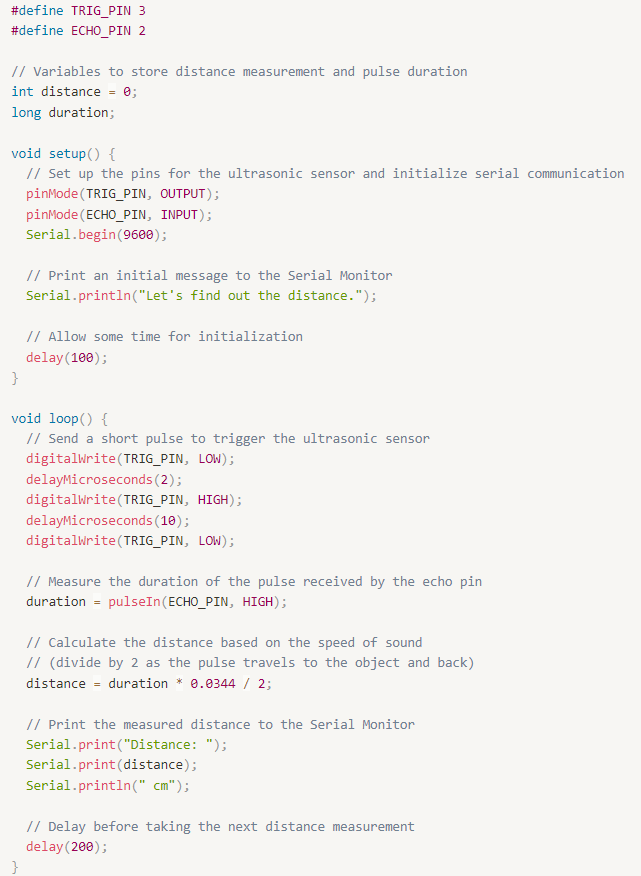
**EXPERIMENT 5: Interfacing with Ultrasonic Sensor**

**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 code, open the Serial Monitor in the Arduino IDE.
   * Set 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:**

****