

TEXAS A&M UNIVERSITY
ROBOTICS TEAM & LEADERSHIP EXPERIENCE

Programming and Git/GitHub

TURTLE Hatchling

VS Code



Integrated Development Environment (IDE)

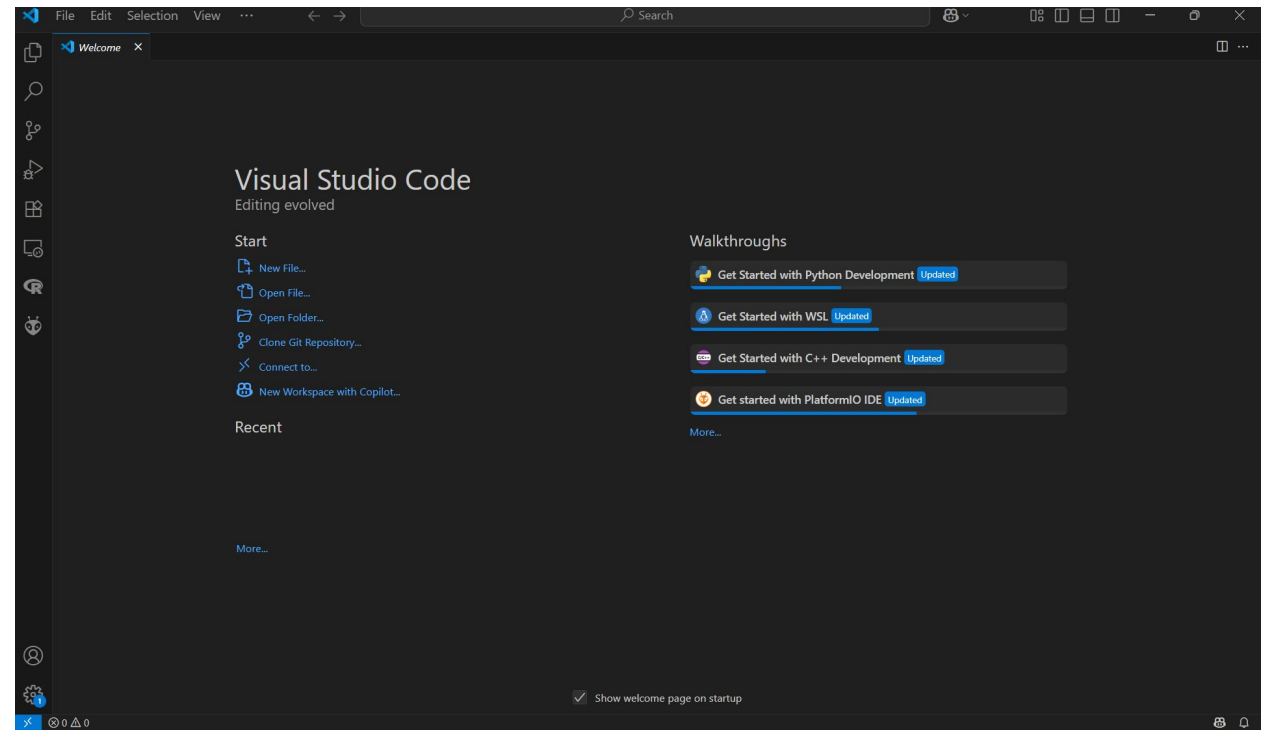
Powerful, popular, and customizable

Git and GitHub integration

Versatile across languages and types of project

Microsoft Store or

<https://code.visualstudio.com/>



Extensions to add

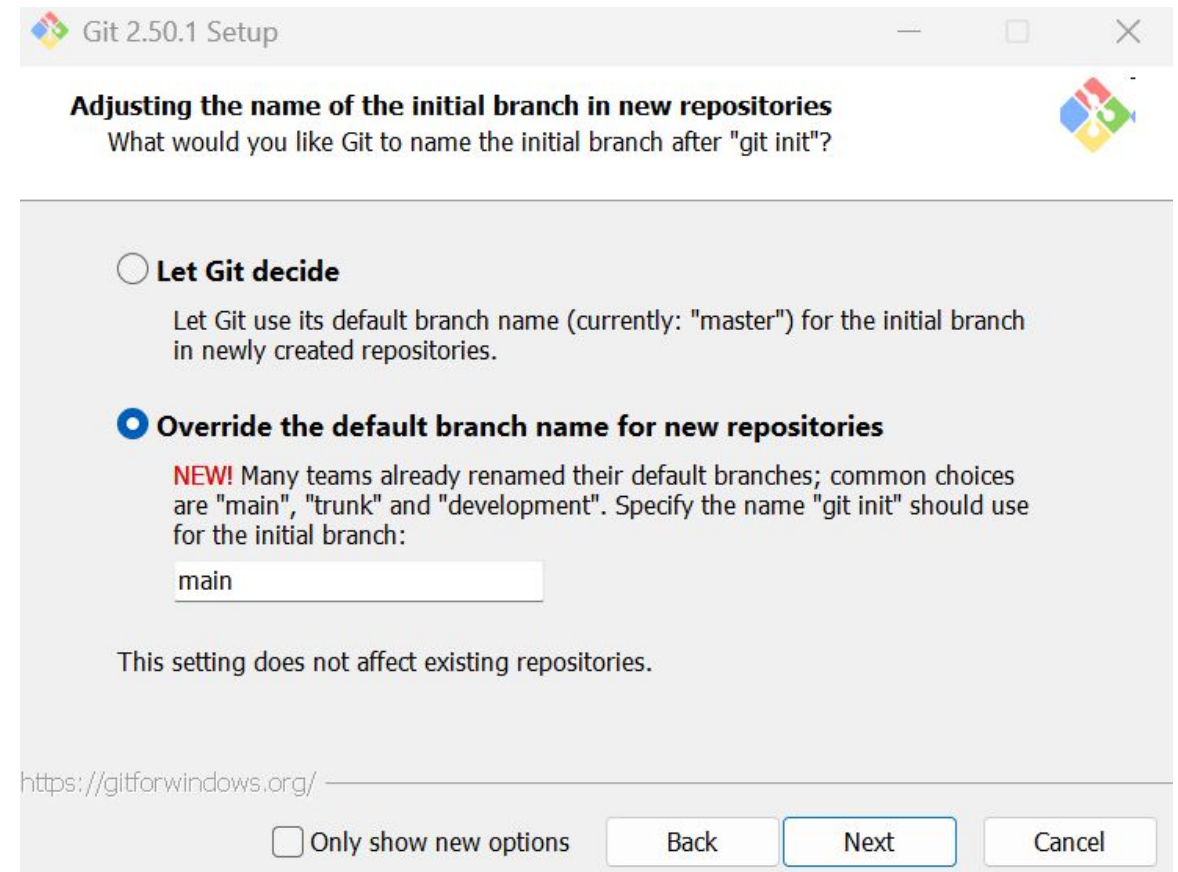
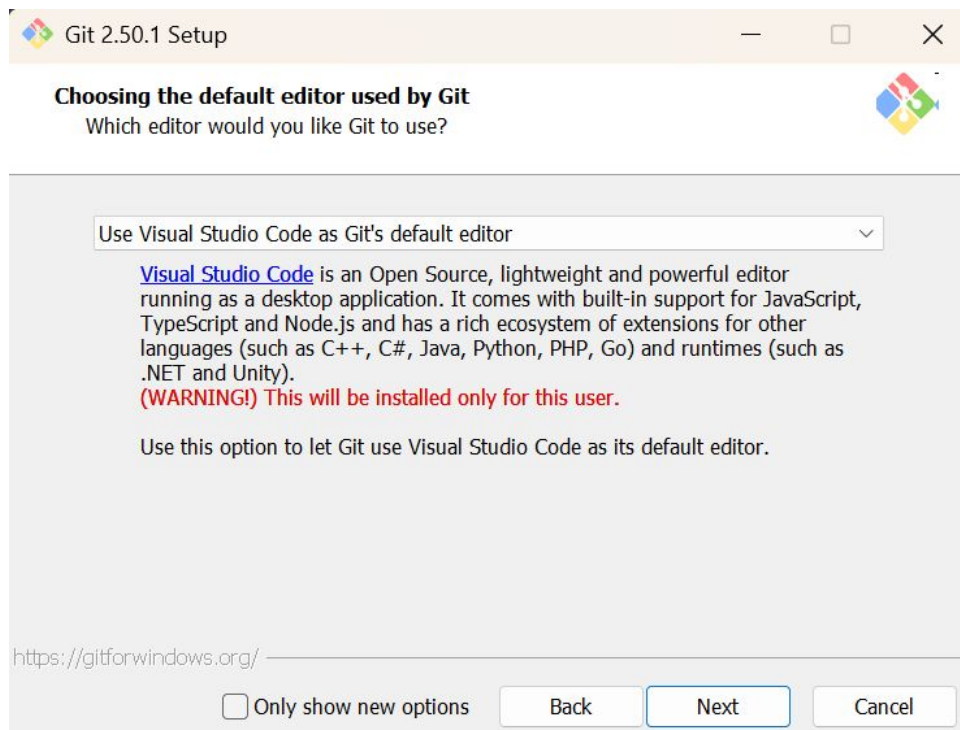


- C/C++ Extension Pack by Microsoft
 - Adds syntax highlighting
- PlatformIO IDE
 - Allows interfacing with ESP32
- Prettier (Personal preference)
 - Code formatter that makes code more legible
- GitHub CoPilot (Be careful to avoid using this for class)
 - AI chat and coding assist that scans files and autocompletes

Installing Git



- Go to this website to install Git:
 - <https://git-scm.com/downloads>
- Use the defaults on every step except the name of the “master” branch. Change the name to “main”



Github Account and Git Setup



1. Create a GitHub account

<https://github.com/>

2. Configure Git:

Open a terminal (or command prompt)

Set your username: `git config --global user.name "<First Last Name>"`

Set your email address: `git config --global user.email "<your@email.com>"`

3. Create a Repository on GitHub:

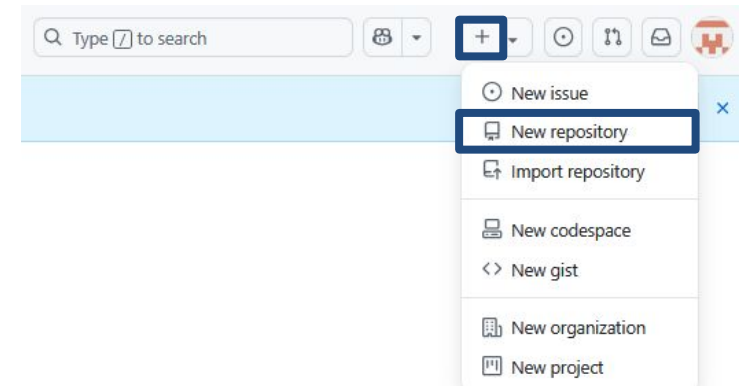
Click the "+" icon in the top right corner and select "New repository"

Give your repository a name and an optional description

Choose a repository visibility (public or private)

Click "Create repository"

Note: Hatchling uses phrases in angled brackets `<>` as placeholders. I.e. someone named John Doe would put "John Doe" instead of "`<name>`"

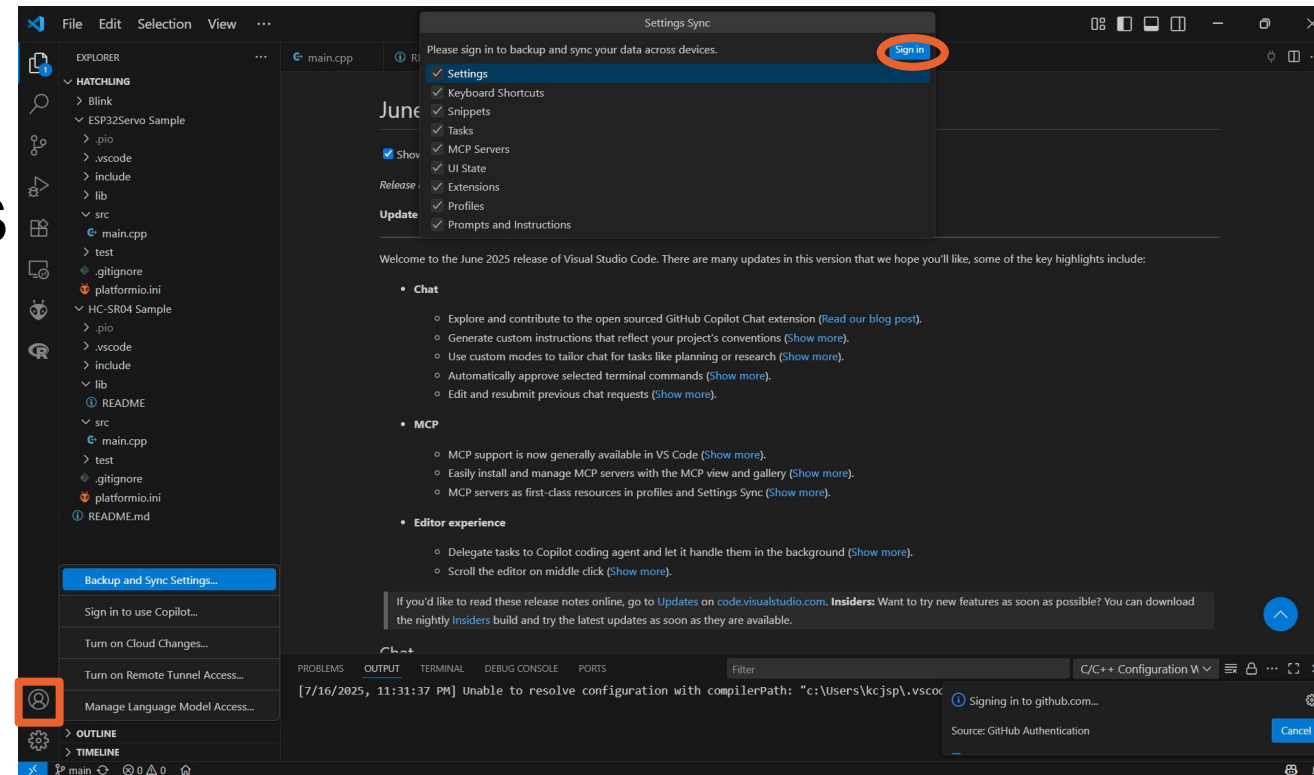


Adding GitHub to VS Code



1. Click “Accounts”
2. Click “Backup and Sync Settings”
3. Click “Sign In”
4. Click “Sign in with GitHub
5. Authorize account access
6. Confirm by clicking “Accounts”

Note: Git is required to be installed. Git commands will work in the integrated terminal



ESP32 Communication Driver

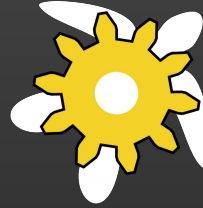


Silicon Labs CP210x USB to UART Bridge must be installed for your laptop to communicate with the ESP32

Linux: `sudo adduser <username> dialout`

Windows: Google "[CP210x](#)"

Most will download "CP210x Windows Drivers" and run "CP210xVCPInstaller_x64.exe"



TEXAS A&M UNIVERSITY
ROBOTICS TEAM & LEADERSHIP EXPERIENCE

Git and GitHub

Why use Git/GitHub?



Git/GitHub are incredibly powerful tools. Think about a Google Doc for your code but on steroids. You can:



- Have multiple people working offline and sync everyone's changes later seamlessly
- Quickly swap between different iterations of your code with a single command
- Develop multiple new features independently of each other using branches



What is Git?



- Standalone Local software that is either already installed (Linux and Mac) or must be installed (Windows)
- Allows you to track changes and work on projects offline
- Can commit new changes or revert to previous versions
- Base system software that is used as a tool by Github



Why is Git Important?



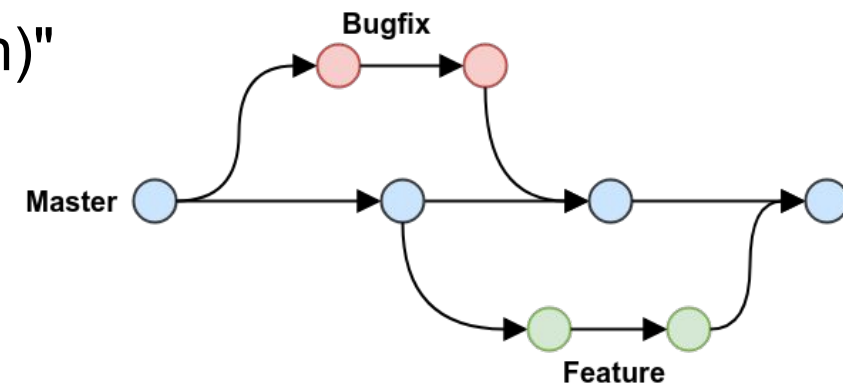
- Git is the tool used by GitHub and is what powers it
- The base software that allows you to make repositories, branches, save (commit) progress, and revert to previous versions of repositories
- Allows for offline project management through various branches and repositories



What are Branches?



- A duplicate of the code to change which you can personalize and NOT affect the original main branch
- Changes in one branch don't affect other branches
- Allows you to make changes in code and if you mess up at any point and break the app etc, you can simply delete branch and get back at main
- Use it to not mess up main branch: (Unbroken branch, not updated unless absolutely clear new code is good)
- When you get a good branch made, you can MERGE into the main branch and then delete the forked branch
- "SirTURLE merged 1 commit into main from auth (branch)"



Basics Commands of Git



Initialize a new repository:

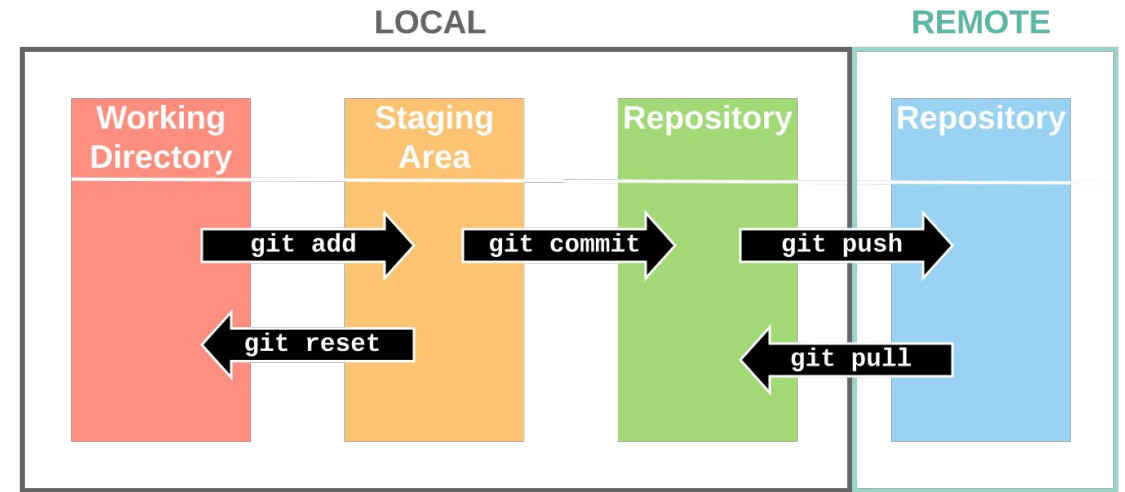
- “git init”

Save Progress:

- “git add .”
 - This adds all unsaved progress
- Can specify which file to add as well
 - “git add <file name>”
- Puts specified files into the staging process to then be committed

Commit Changes:

- “git commit -m “<describe changes>”



More Git Commands



Checkout

- Allows you to create new branches and go back to previous versions of your repositories
- "git checkout -b <name of branch>"
 - Creates a new branch
 - Once created, omit the "-b" to switch between branches
- "git checkout <hashcode>"
 - Can be used to go to previous versions of a branch

Merge

- Used to sync all changes made in one branch to another one
- "git merge <branch receiving the changes> <branch supplying the changes>"
 - "git merge main hotfix_branch"
 - This would update your main branch with all the fixes you'd made in hotfix_branch

More Git commands



Branch

- Used for various tasks in managing branches
- “git branch -d <branch name>”
 - Deletes a branch, usually done once its been merged to main

See the status of your repository:

- “git status”
 - Total overview with uncommitted files, current branch, etc.
- “git log”
 - Only outputs a log of all commits made

GitHub Commands



Clone:

- “git clone <repository link>”
- This clones the repository to your current directory

Push Changes:

- “git push origin main”
 - Pushes changes from local branch to remote repository (called “origin” by default)

Pull Submitted Changes:

- “git pull origin main”
- Pulls changes from the main branch of the remote repository (named “origin”) and merges them into your current local branch

What is GitHub?



- The online version of Git
- Web-based platform which uses and relies on Git
- Allows you to collaborate with others and pull from their projects
- Can be accessed through VSCode as well as through the terminal

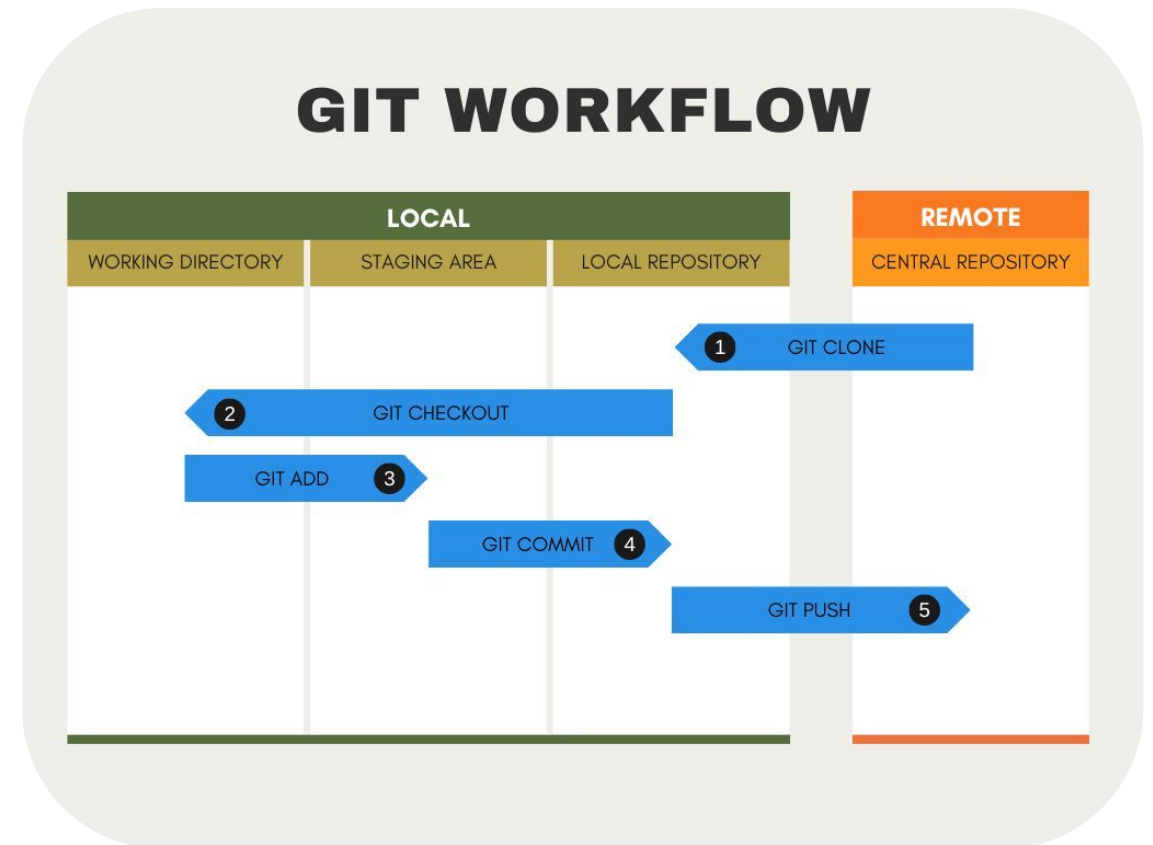


GitHub Basics



GitHub basically works by having a remote copy of your repository stored on its servers.

- Remote repository is called “origin” in any git commands you use
- To sync changes between your local and remote repositories, we use the “git push” command
 - Typically looks something like “git push origin *branch_name*”



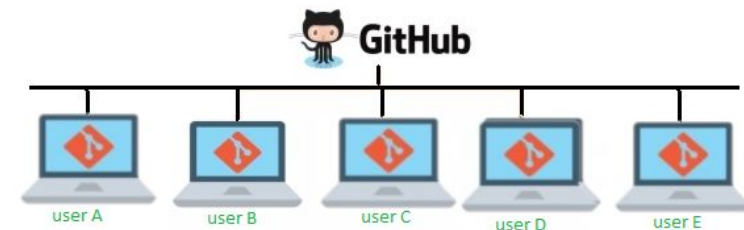
What is GitHub's Application?



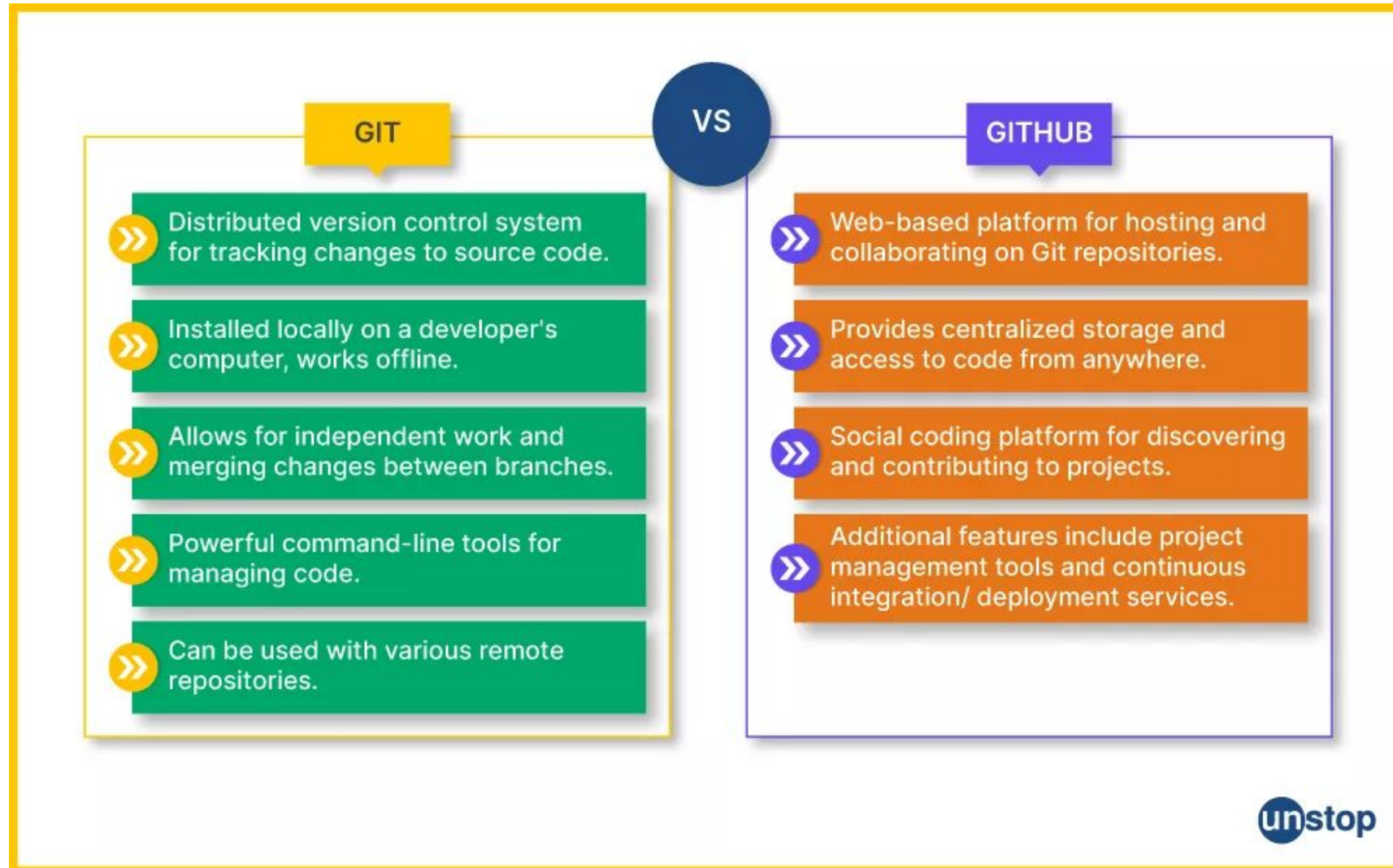
- Used for collaborative programming
- Utilized worldwide in business environments
- Facilitated Teamwork

Where is this applied:

- In advanced projects, GitHub is used to help organize projects and files, and collaborate on deliverables
- In Hatchling, GitHub can be used with your teams to program your competition robots



Git vs GitHub





GitHub Setup (one or both)

Command Line

- Allows for access to GitHub from anywhere on your computer using the command line
- Easy way is with GitHub CLI
- Download and install from website
- `gh auth login`
- `winget install --id GitHub.cli`

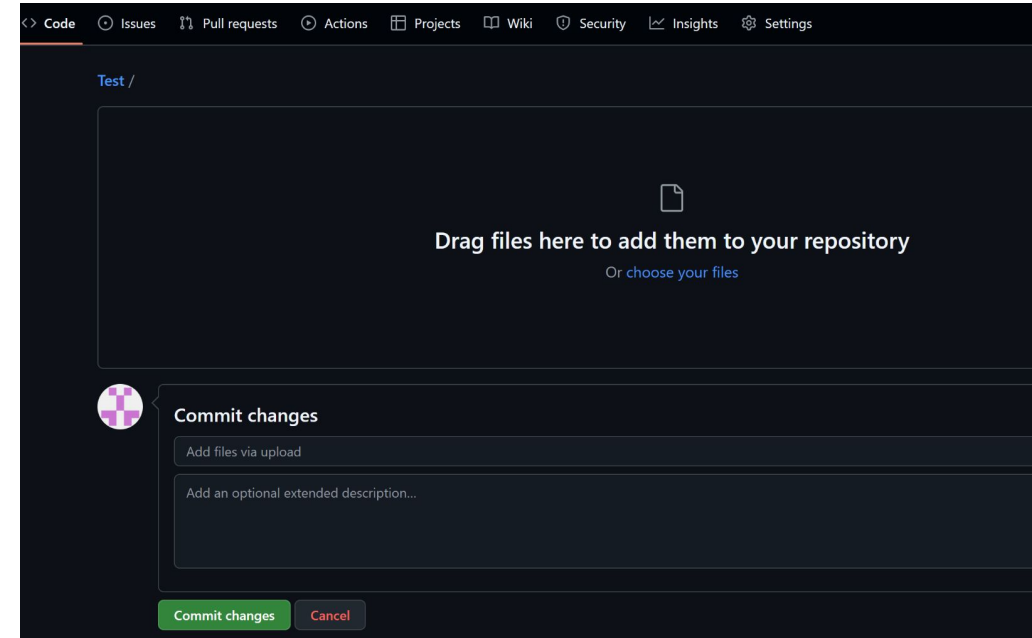
VS Code Integration

- Only works inside VS Code
- On the left side bar press Accounts
- Login to Github with your browser
- This is what I use normally

GitHub Method 1 (Existing Code)



- Inside your project press “Initialize Repository”
- Stage and commit changes with a relevant commit message
- Login to GitHub and create a new repository
 - DO NOT make a README
- Copy and paste the contents of “...or push an existing repository from the command line” to the project terminal and run them
- Refresh GitHub to check that it worked



GitHub Method 2 (Before Code)



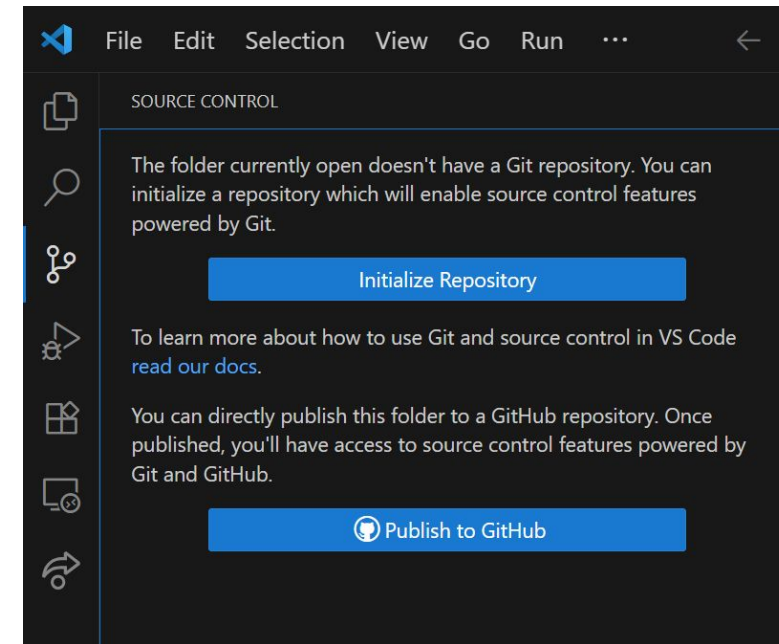
- Login to GitHub and create a new repository
 - README is optional for now. You should make one eventually to explain your projects
- Run 'git clone <https copied from GitHub repo>' in any terminal navigated to your storage location
- Refresh GitHub to check that it worked
- cd and add files as needed

```
C:\Users\nwmer>git clone https://github.com/NathanMer/Test
Cloning into 'Test'...
warning: You appear to have cloned an empty repository.
```

GitHub VSCode (Before Code, Source Control)



- Click “Initialize Repository” under Source Control
- Stage changes
- Commit changes through Source Code
- Can also select “Publish to GitHub” to put the opened file into a repository.



GitHub VSCode (Before Code, Terminal)



- VSCode setup method:
 - Click Accounts on left side bar and sign into GitHub
 - Run 'git clone <https copied from GitHub repo>' in VS Code terminal navigated to your storage location
- Refresh GitHub to check that it worked
- Create a new folder, initialize git, add files, and commit changes

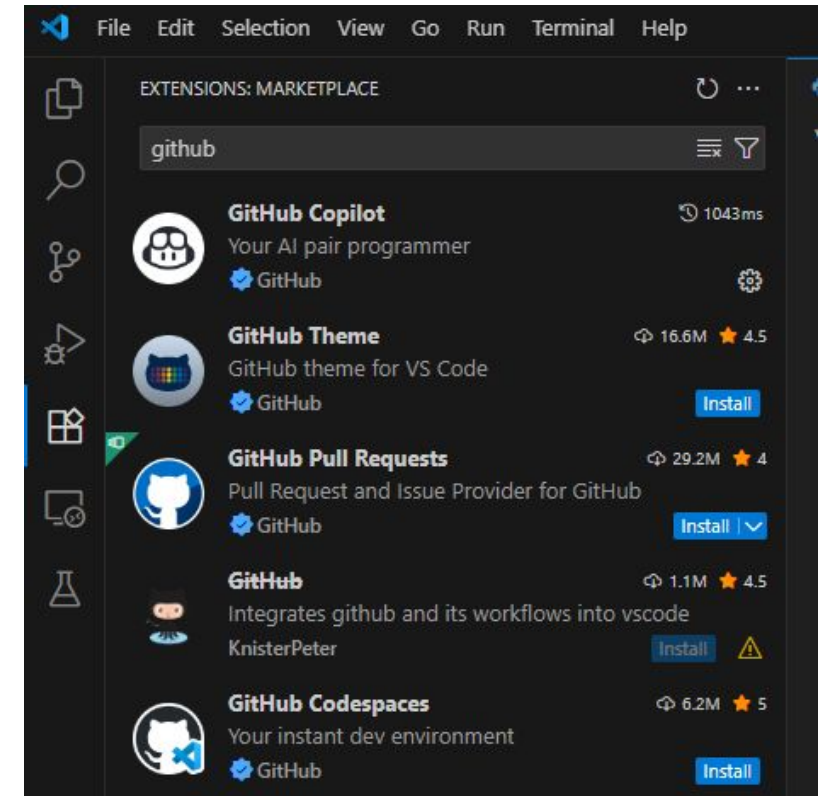
A screenshot of the Visual Studio Code terminal window. The terminal title bar shows 'pwsh' and a warning icon. The terminal content shows a PowerShell prompt at 'C:\Users\nwmer\Desktop\Coding\Test' where the command 'git clone https://github.com/NathanMer/Test' has been executed. The output shows 'Cloning into 'Test'...' followed by a warning: 'warning: You appear to have cloned an empty repository.' The prompt is now ready for the next command.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\nwmer\Desktop\Coding\Test> git clone https://github.com/NathanMer/Test
Cloning into 'Test'...
warning: You appear to have cloned an empty repository.
PS C:\Users\nwmer\Desktop\Coding\Test>
```

VS Code Extensions



- Along with GitHub Copilot, many GitHub extensions are available on VS Code which makes it (in my opinion) easier to use. Extensions such as “GitHub Repositories” make GitHub navigation easier.
 - EX: GitHub Repositories allows you to access and easily switch to repositories linked onto your GitHub account



Commitment Issues



- GitHub does not automatically sync changes
- Code must be pulled and pushed to your remote version
- Before you push
 - Add your files to **commit** (Called stage in VS Code)
 - 'git add <files>' or 'git add *' (* adds all tracked files)
 - Commit your changes that you want to push
 - 'git commit -m "<message>"' (write a descriptive commit message)
- You can use 'git status' to make sure there won't be issues first
 - These can usually be avoided with 'git stash' and 'git stash pop'
 - The merge editor in VS Code also works

Helpful Links:



- How to use a terminal:
 - Linux: <https://terminalcheatsheet.com/>
 - Mac: <https://support.apple.com/guide/terminal/welcome/mac>
 - Windows (PS): <https://www.pdq.com/powershell/>
- GitHub:
 - <https://github.com/>
- GitHub & Git Tutorials:
 - <https://youtu.be/mJ-qvsxPHpY?si=UorINq0JFX33fm5z>
 - <https://youtu.be/a9u2yZvsqHA?si=-rjfo1pnPLrxPvJH>
 - https://youtu.be/USjZcfj8yxE?si=k_Al8ScCOkn90lwF
- Setup GitHub through VSCode
 - <https://youtu.be/z5jZ9lrSpqk?si=LU47SwwwvsZwUebmN>



TEXAS A&M UNIVERSITY
ROBOTICS TEAM & LEADERSHIP EXPERIENCE

Let's Code

Three Example Programs



- Blink
 - Simple led blinking code to test PlatformIO IDE VS Code extension setup and ESP32-Wroom-32D interfacing.
- ESP32Servo
 - Simple servo movement code to practice looking up datasheets
- HC-SR04
 - Simple distance displaying code to demonstrate sensors implementation.

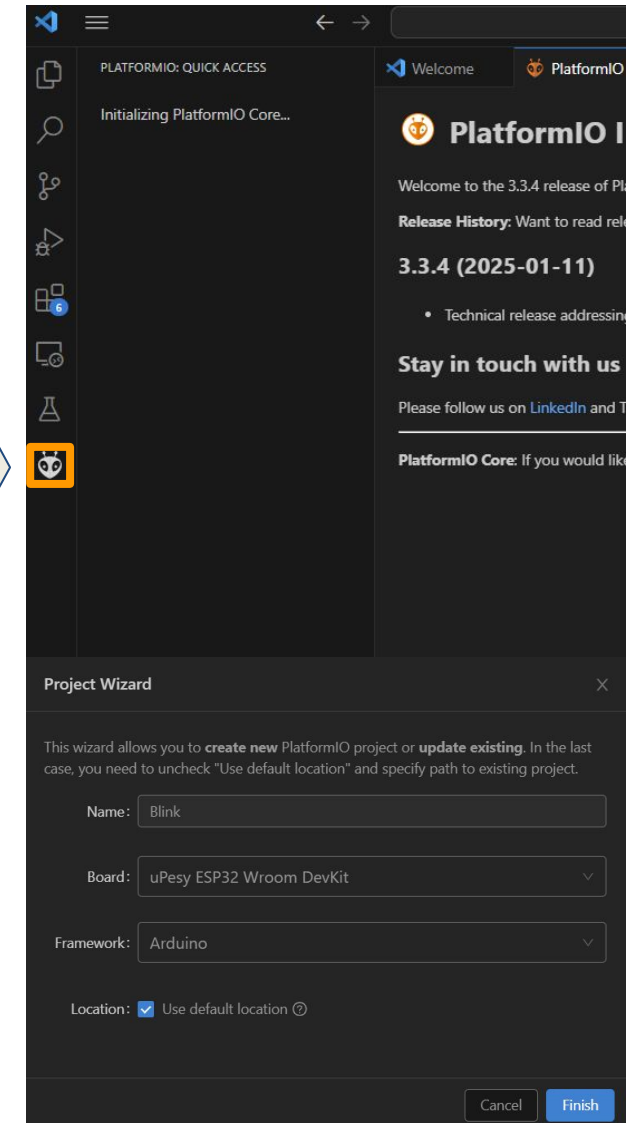
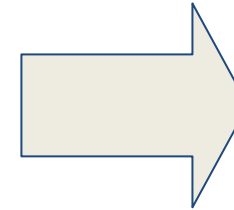
Example codes can be found at

<https://github.com/turtle-robotics/Hatchling-Examples>

New Project



- Click the PlatformIO logo in the activity bar (left)
- QUICK ACCESS > PIO Home > Open > New Project
- Name: Blink
- Board: uPesy ESP32 Wroom DevKit
- Framework: Arduino
- Location: <place to store code that you'll remember>



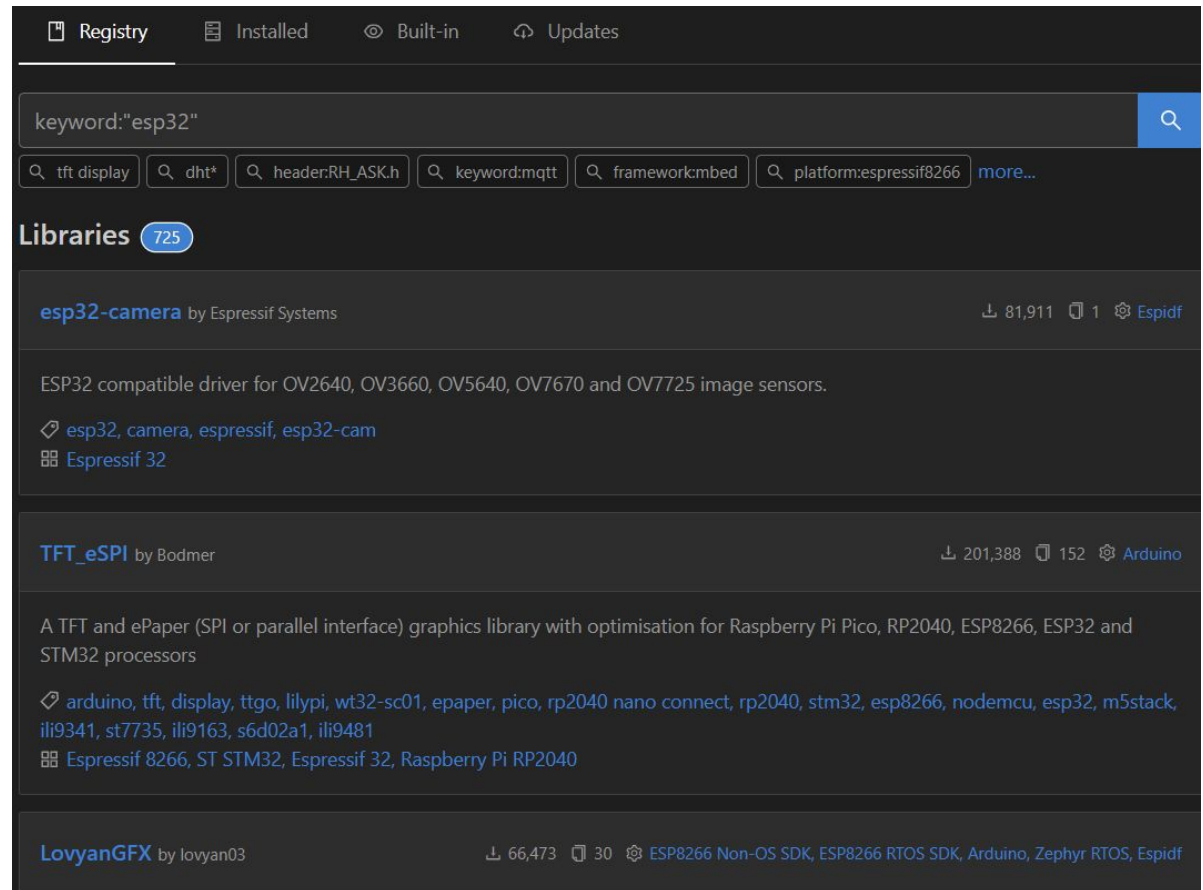
Note: It may take a minute to load. If stuck, restart VS Code

Code Libraries



- More complicated electronics like servos or sensors need more code to send/receive their data
- Libraries will have documentation that explains how to use them

Note: Like COTS Cad files, someone's probably already did this work for you!



Code Libraries



Code libraries are pre-written modules of code, often containing useful classes/functions

- Basically, you use someone else's code to do common tasks you don't want to re-write code for

To use a code library, we “import” it at the beginning of our code.

- PlatformIO automatically writes in the Arduino library

Importing in C++



Key-word for importing - `#include`

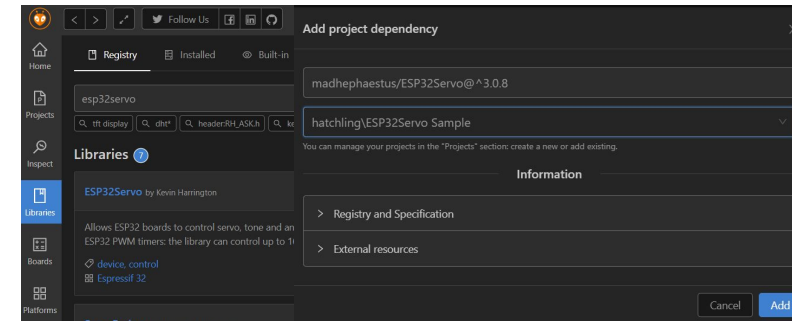
The Arduino library is where all your reading, writing, serial printing, etc. functions are coming from

```
1  #include <Arduino.h> // imports the Arduino library, contains stuff like digitalWrite() and digitalRead()
2
3  void setup() {
4  |  // put your setup code here, to run once:
5  |
6  |
7  void loop() {
8  |  // put your main code here, to run repeatedly:
9  |
10 |
```

Downloading New Libraries



- Click the PlatformIO logo in the activity bar (left)
- QUICK ACCESS > PIO Home > Libraries
- Name: ESP32Servo
- Add to Project



Documentation is your best resource for using any libraries you import

pls pls pls read documentation

platformio.ini



```
platformio.ini •
ESP32Servo Sample > platformio.ini
1  ; PlatformIO Project Configuration File
2  ;
3  ; Build options: build flags, source filter
4  ; Upload options: custom upload port, speed and extra flags
5  ; Library options: dependencies, extra library storages
6  ; Advanced options: extra scripting
7  ;
8  ; Please visit documentation for the other options and examples
9  ; https://docs.platformio.org/page/projectconf.html
10
11  [env:upesy_wroom]
12  platform = espressif32 ; auto
13  board = upesy_wroom ; auto
14  framework = arduino ; auto
15  lib_deps = madhephaestus/ESP32Servo@^3.0.8 ; auto
16  monitor_speed = 115200 ; Specific to the board make sure this matches the main.cpp file
```

Note: lib_deps will auto populate when importing libraries through PlatformIO

SG90 Servo Datasheet Lookup



When we look at the documentation for our servo library, we find that it asks us to provide the minimum and maximum pulse width of the servo.

To find values like these, we must consult the documentation at:

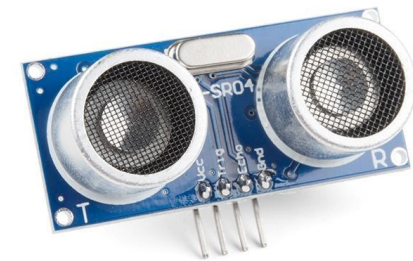
http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sq90_datasheet.pdf

Ultrasonic Sensor (HC-SR04) Example



Next, use the HC-SR04 distance sensor

- Create a new project
- Click the PlatformIO logo in the activity bar (left)
- QUICK ACCESS > PIO Home > Libraries
- Name: HC-SR04
- Add to Project



HC-SR04 Datasheet Lookup



The library for the HC-SR04 isn't asking for any specs. Do we still have to look at the datasheet?

YES!!!!

Even if the library isn't asking for anything directly, you should always check the datasheet to learn things like the operating voltage of the device, pinout, specifications on the distance measurable, etc. We can find these here:

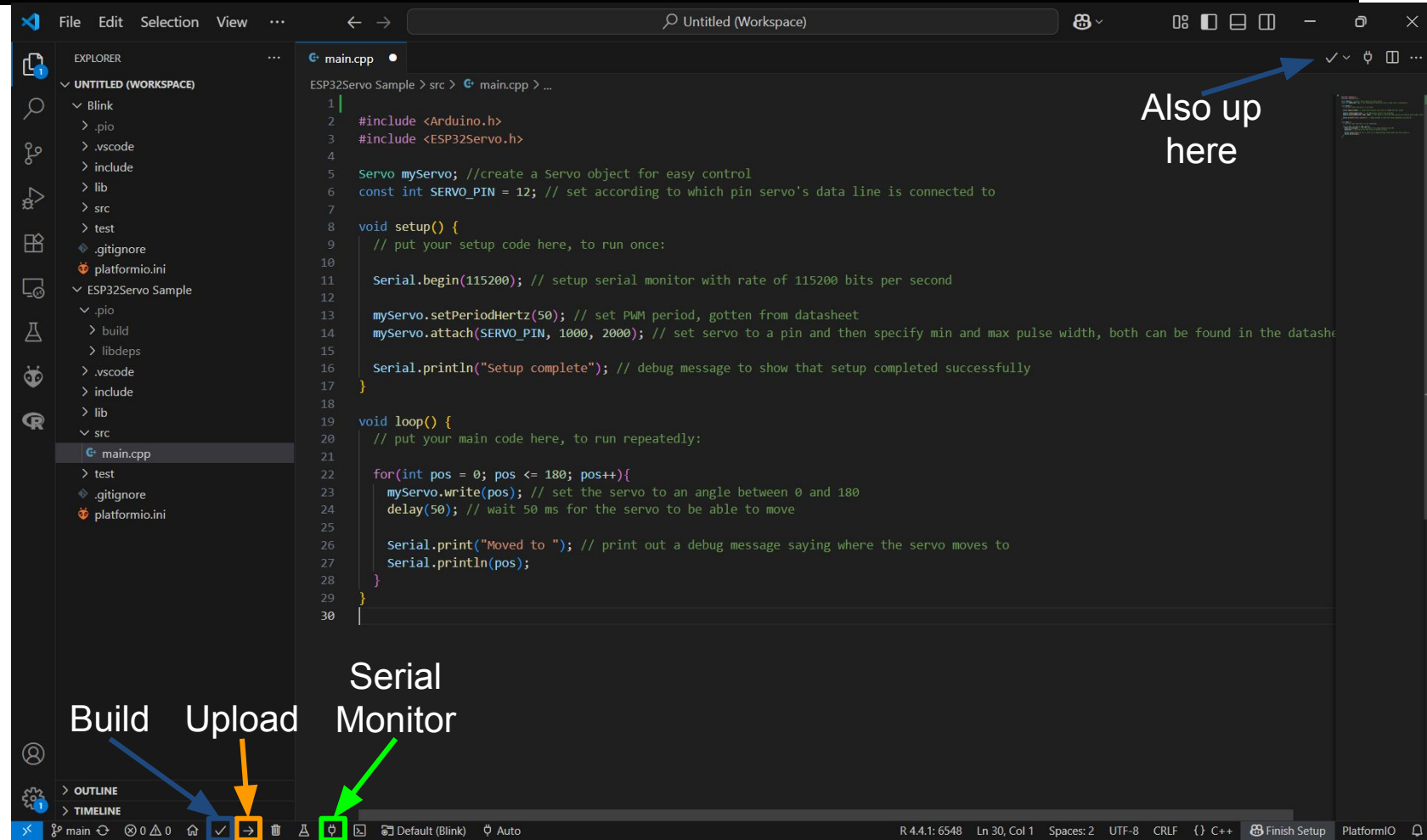
<https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HC-SR04.pdf>

Build and Upload Code



Build: Checks for syntax errors then compiles your code into a MCU readable machine language.

Upload: Transfers compiled code to MCU



If your ESP32 is doing what you intended, congrats!

If not, you have some troubleshooting to do...

Next Milestone



Milestone: Prototype Review

Date: Week 9 - Prototype (2 week from today)

Expectation: Have a CAD assembly of a drive system. Have a finished electronics wiring diagram.

Exceed Expectation: Have a CAD of the entire robot and started prototyping. Began programming the robot.


Impact: We will review design viability and suggest improvements. Potential to prototype your mechanism.



TEXAS A&M UNIVERSITY
ROBOTICS TEAM & LEADERSHIP EXPERIENCE

Electronics and Soldering!

Next Week



“Machines take me by surprise with great frequency.”

~ Alan Turing



Hatchling

