TEXAS A&M UNIVERSITY
ROBOTICS TEAM & LEADERSHIP EXPERIENCE

Programming and Git/GitHub

TURTLE Hatchling

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:

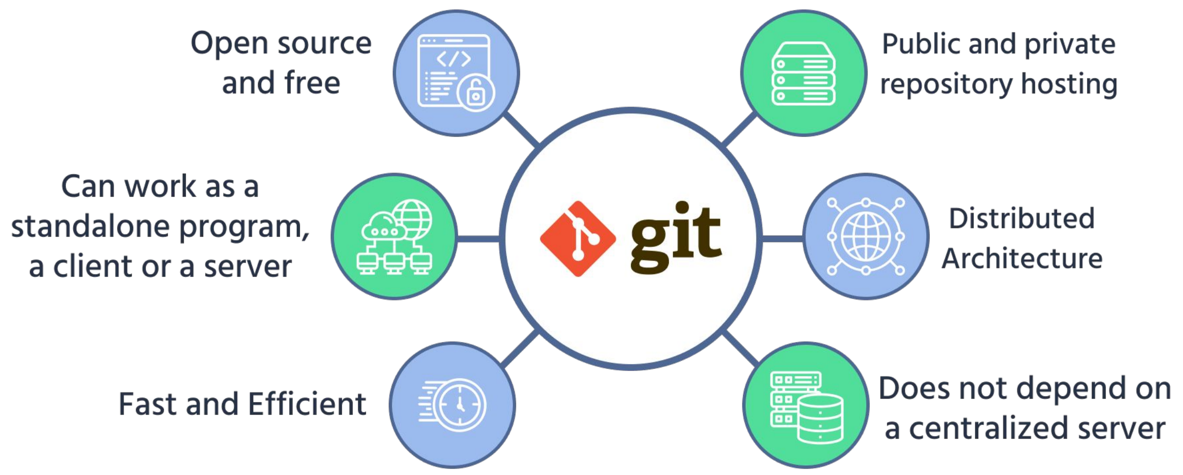
- Have multiple people working offline and sync everyone's changes later seamlessly
- Quickly swap between different iterations of your code with a single command
- Develop multiple new features independently of each other using branches

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

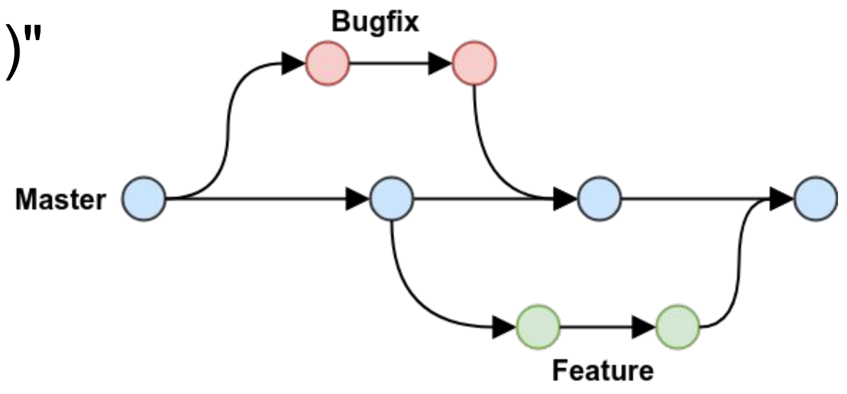
Why is Git Important?

- 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


Open source and free
Public and private repository hosting
Can work as a standalone program, a client or a server
git
Distributed Architecture
Fast and Efficient
Does not depend on a centralized server

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:

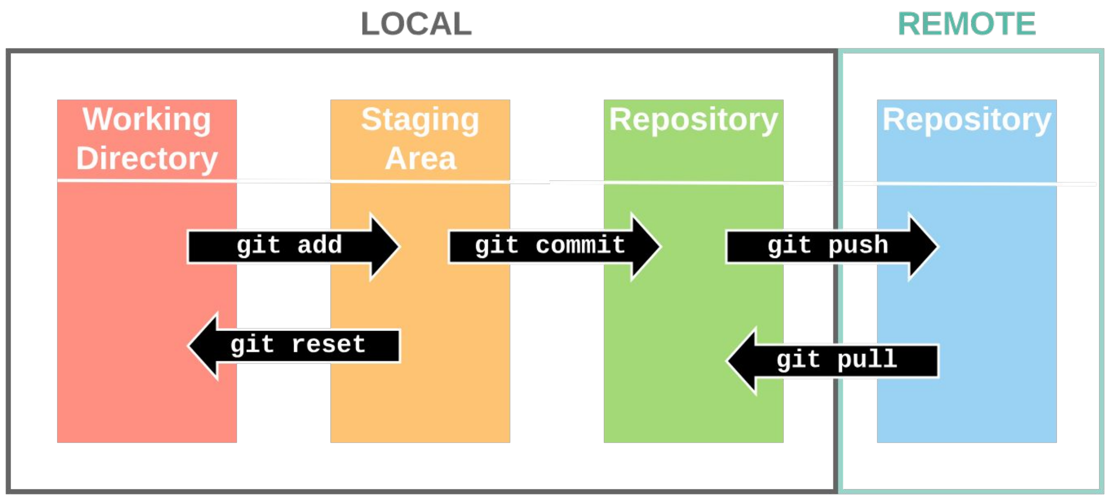
- “git init”

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

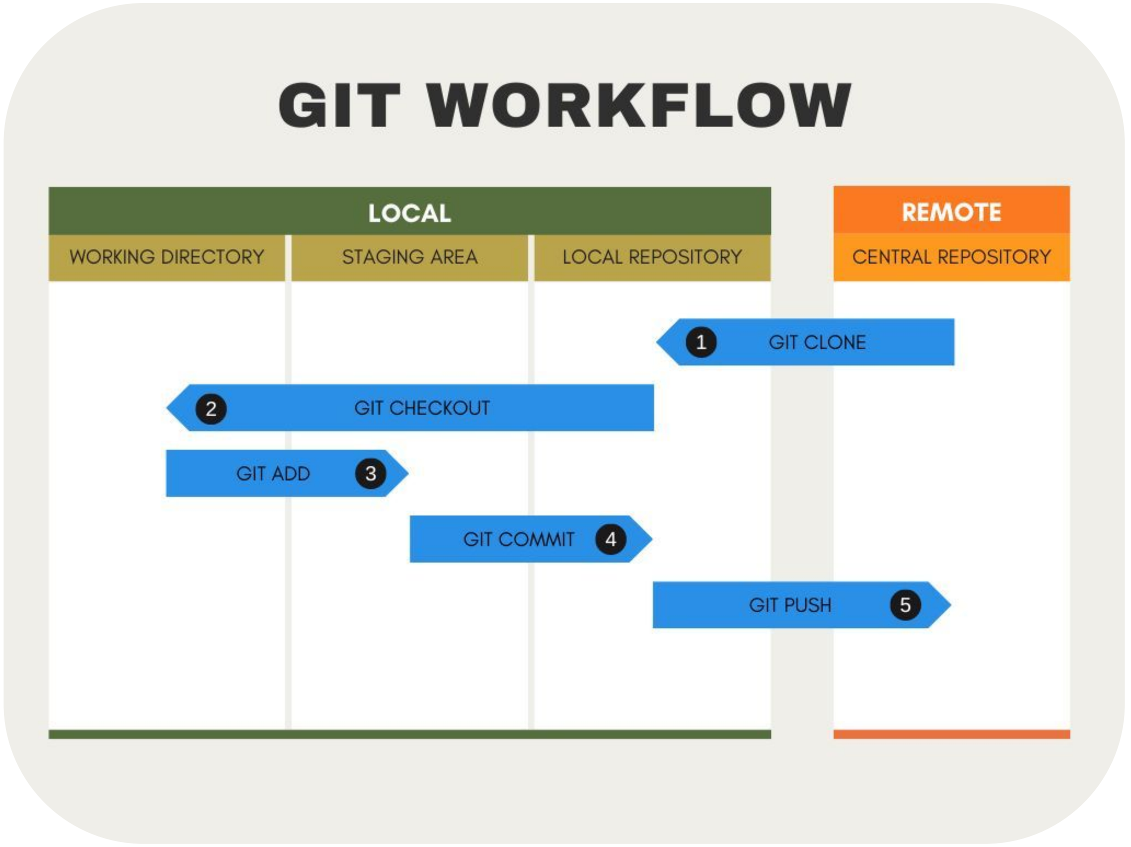
What is GitHub?

- 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*"

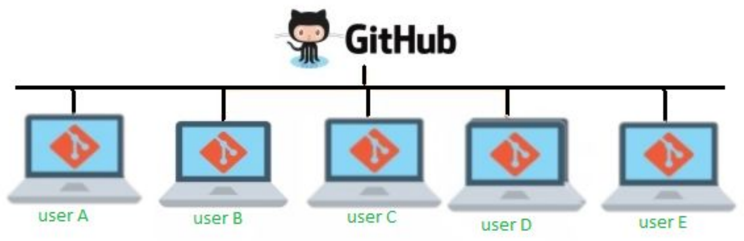


What is GitHub's Application?

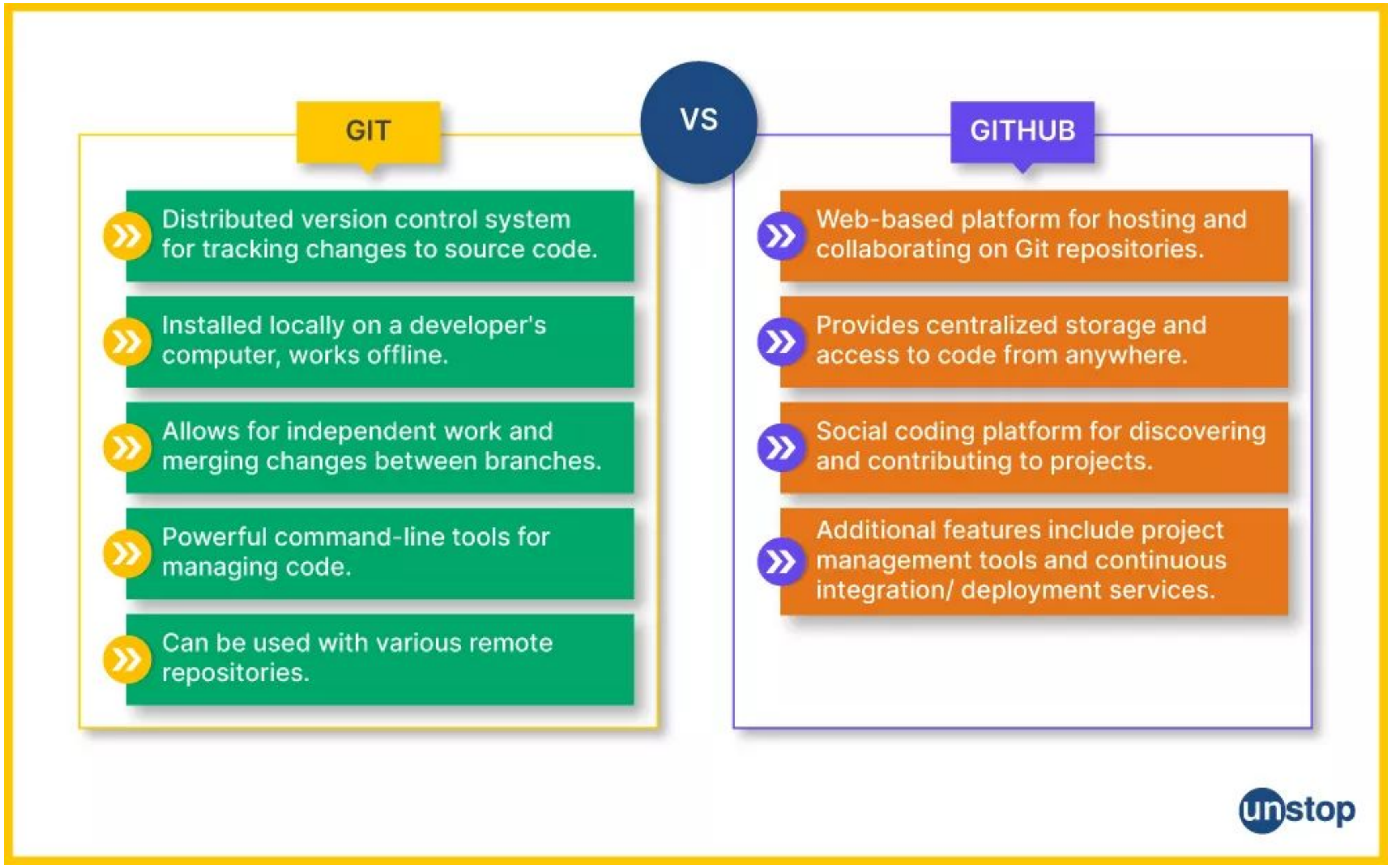
- Used for collaborative programming
- Utilized worldwide in business environments
- Facilitated Teamwork

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



VS

GIT

- Distributed version control system for tracking changes to source code.
- Installed locally on a developer's computer, works offline.
- Allows for independent work and merging changes between branches.
- Powerful command-line tools for managing code.
- Can be used with various remote repositories.

GITHUB

- Web-based platform for hosting and collaborating on Git repositories.
- Provides centralized storage and access to code from anywhere.
- Social coding platform for discovering and contributing to projects.
- Additional features include project management tools and continuous integration/ deployment services.

unstop

GitHub Setup (one or both)

Command Line

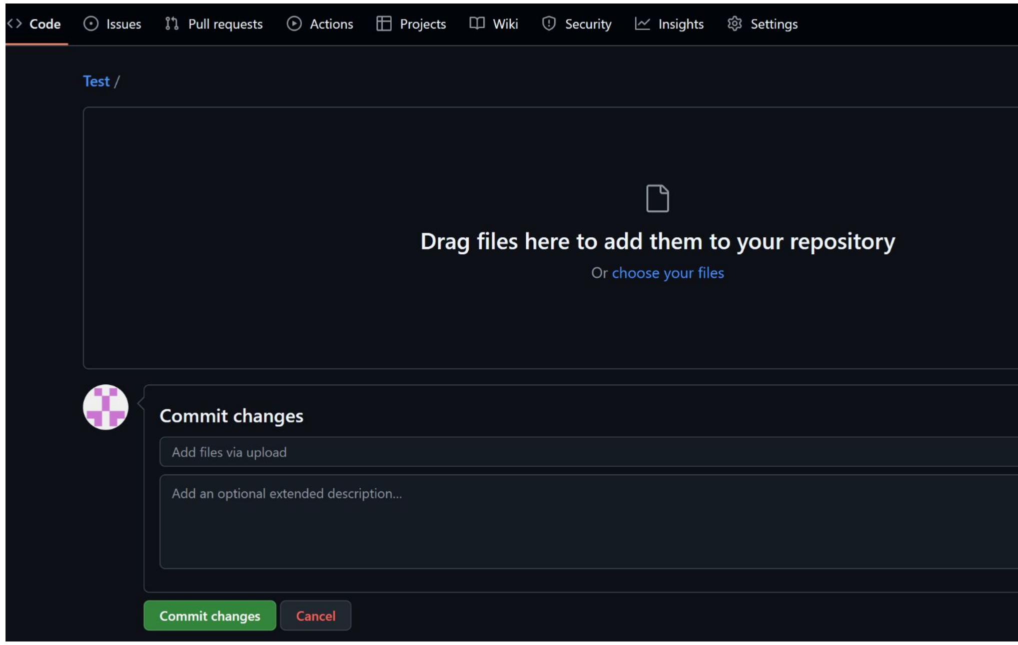
- Allows for access to GitHub from anywhere on your computer using the command line
- Easy way is with GitHub CLI
- Download and install from website
- gh auth login
- winget install --id GitHub.cli

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



Drag files here to add them to your repository

Or choose your files

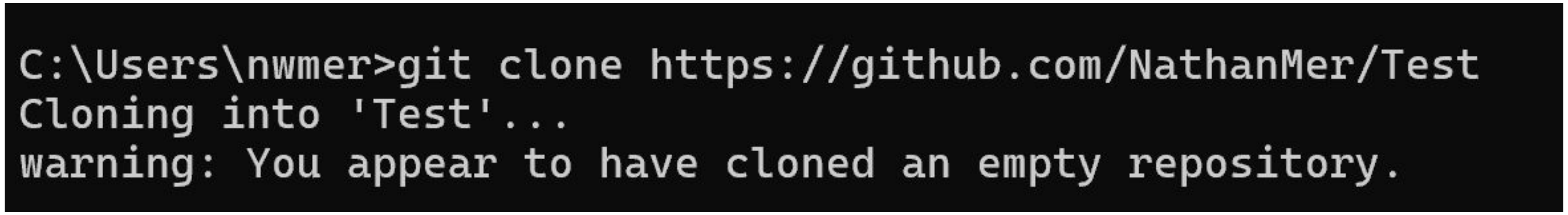
Commit changes

Add files via upload

Add an optional extended description...

GitHub Method 2 (Before Code)

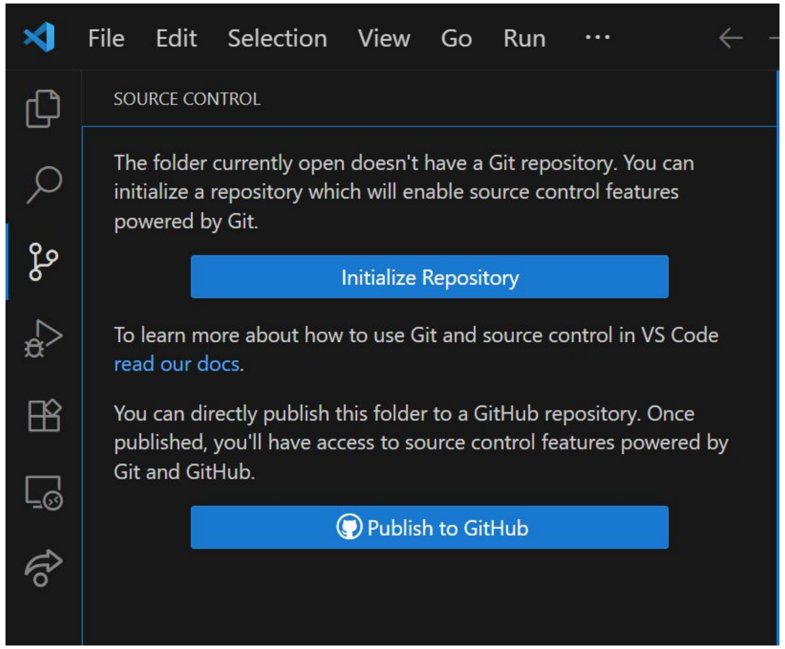
- Login to GitHub and create a new repository
  - README is optional for now. You should make one eventually to explain your projects
- Run 'git clone <https copied from GitHub repo>' in any terminal navigated to your storage location
  - Refresh GitHub to check that it worked
  - cd and add files as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



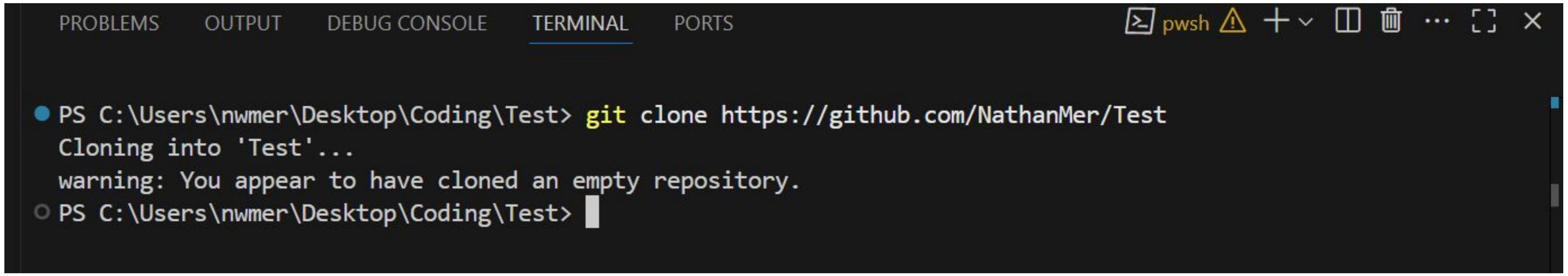
The folder currently open doesn't have a Git repository. You can initialize a repository which will enable source control features powered by Git.

To learn more about how to use Git and source control in VS Code read our docs.

You can directly publish this folder to a GitHub repository. Once published, you'll have access to source control features powered by Git and GitHub.

GitHub VSCode (Before Code, Terminal)

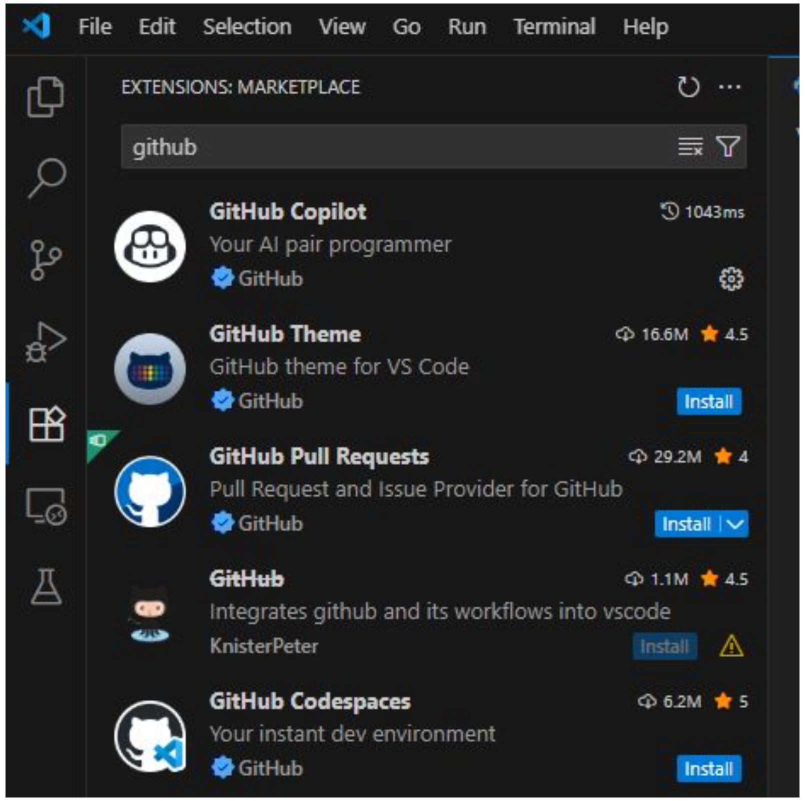
- VSCode setup method:
  - Click Accounts on left side bar and sign into GitHub
  - Run ‘git clone <https copied from GitHub repo>’ in VS Code terminal navigated to your storage location
- Refresh GitHub to check that it worked
- Create a new folder, initialize git, add files, and commit changes



```
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



GitHub Copilot — 1043ms
Your AI pair programmer
GitHub

GitHub Theme — 16.6M ★ 4.5
GitHub theme for VS Code
GitHub

GitHub Pull Requests — 29.2M ★ 4
Pull Request and Issue Provider for GitHub
GitHub

GitHub — 1.1M ★ 4.5
Integrates github and its workflows into vscode
KnisterPeter

GitHub Codespaces — 6.2M ★ 5
Your instant dev environment
GitHub

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=UorlNq0JFX33fm5z
    - https://youtu.be/a9u2yZvsqHA?si=-rjfo1pnPLrxPvJH
    - https://youtu.be/USjZcfj8yxE?si=k_AI8ScCOkn90lwF
- Setup GitHub through VSCode
    - https://youtu.be/z5jZ9IrSpqk?si=LU47SwwvsZwUebmN

TEXAS A&M UNIVERSITY
ROBOTICS TEAM & LEADERSHIP EXPERIENCE

Let’s Code

Three Example Programs

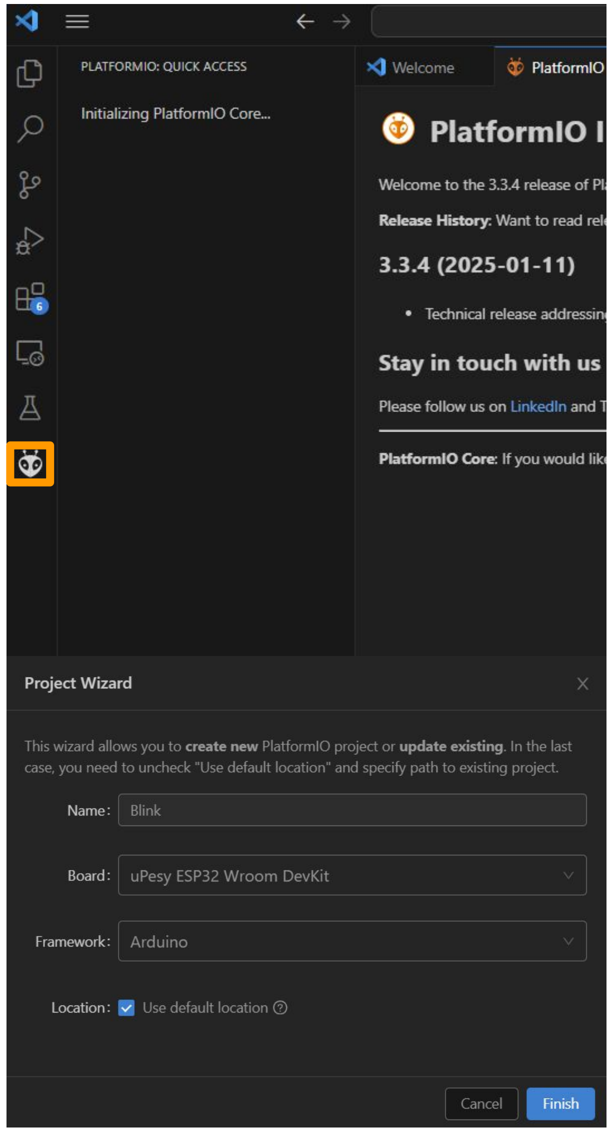
- Blink
  - Simple led blinking code to test PlatformIO IDE VS Code extension setup and ESP32-Wroom-32D interfacing.
- ESP32Servo
  - Simple servo movement code to practice looking up datasheets
- HC-SR04
  - Simple distance displaying code to demonstrate sensors implementation.

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



Initializing PlatformIO Core...

PlatformIO I

Welcome to the 3.3.4 release of Pl

Release History: Want to read rel

3.3.4 (2025-01-11)

- Technical release addressin

Stay in touch with us

Please follow us on LinkedIn and T

PlatformIO Core: If you would lik

Project Wizard

This wizard allows you to **create new** PlatformIO project or **update existing**. In the last case, you need to uncheck "Use default location" and specify path to existing project.

Name: Blink

Board: uPesy ESP32 Wroom DevKit

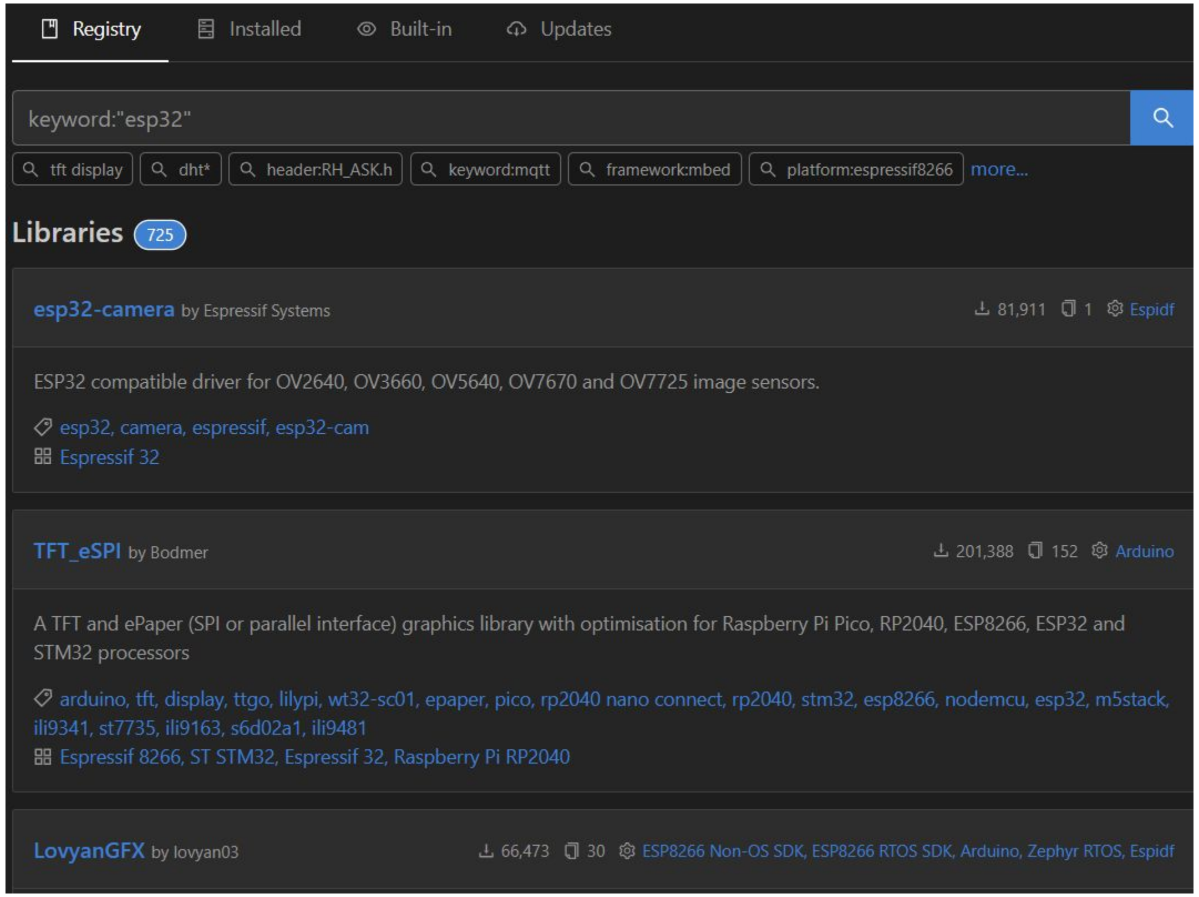
Framework: Arduino

Location: ✓ Use default location

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!



Libraries 725

esp32-camera by Espressif Systems — 81,911 · 1 · Espidf

ESP32 compatible driver for OV2640, OV3660, OV5640, OV7670 and OV7725 image sensors.

esp32, camera, espressif, esp32-cam

Espressif 32

TFT_eSPI by Bodmer — 201,388 · 152 · Arduino

A TFT and ePaper (SPI or parallel interface) graphics library with optimisation for Raspberry Pi Pico, RP2040, ESP8266, ESP32 and STM32 processors

arduino, tft, display, ttgo, lilypi, wt32-sc01, epaper, pico, rp2040 nano connect, rp2040, stm32, esp8266, nodemcu, esp32, m5stack, ili9341, st7735, ili9163, s6d02a1, ili9481

Espressif 8266, ST STM32, Espressif 32, Raspberry Pi RP2040

LovyanGFX by lovyan03 — 66,473 · 30 · ESP8266 Non-OS SDK, ESP8266 RTOS SDK, Arduino, Zephyr RTOS, Espidf

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

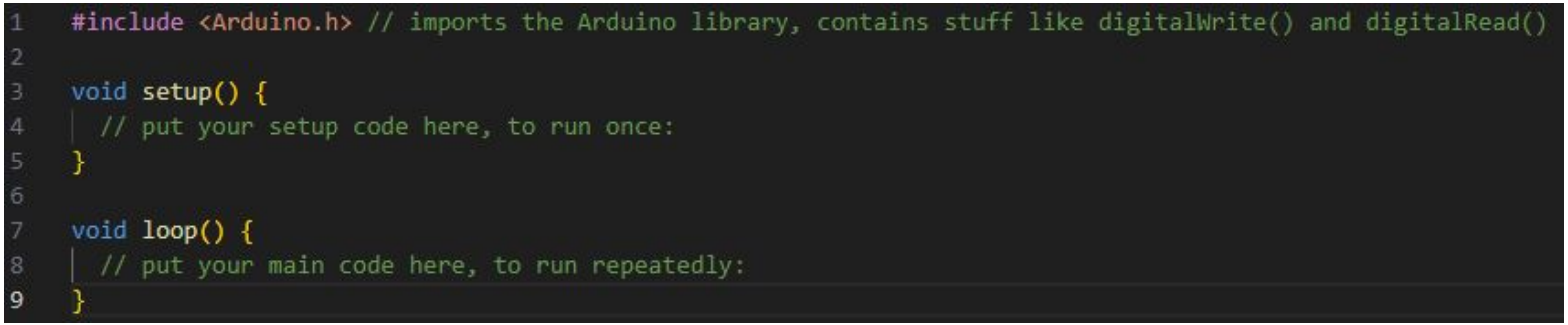
To use a code library, we “import” it at the beginning of our code.

- 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



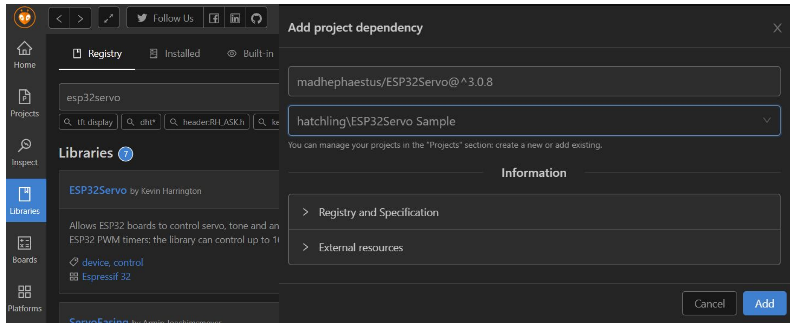
```
#include <Arduino.h> // imports the Arduino library, contains stuff like digitalWrite() and digitalRead()

void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

Downloading New Libraries

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



ESP32Servo by Kevin Harrington

Allows ESP32 boards to control servo, tone and an
ESP32 PWM timers: the library can control up to 1

device, control
Espressif 32

Add project dependency

madhephaestus/ESP32Servo@^3.0.8

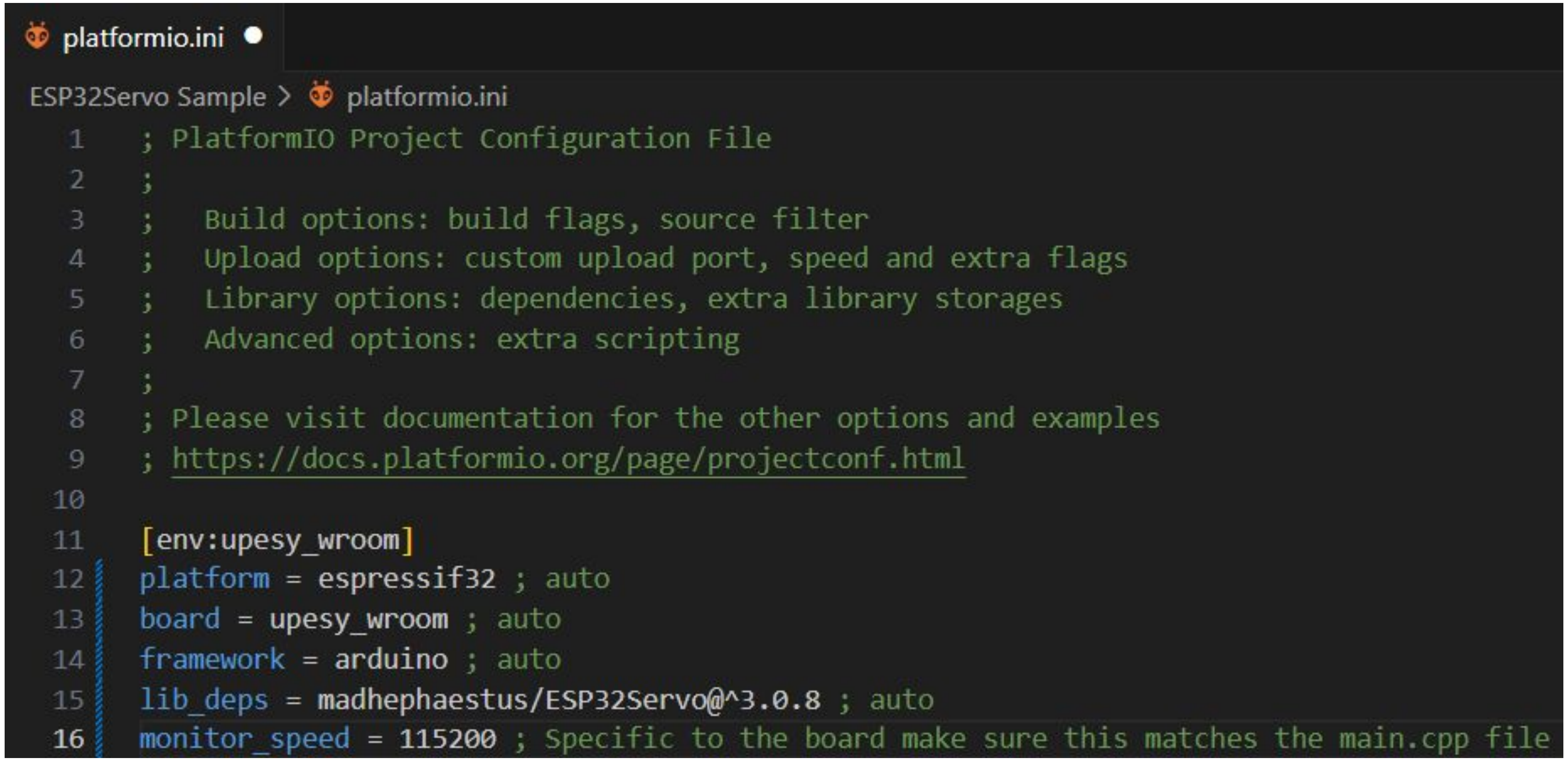
hatchling\ESP32Servo Sample

You can manage your projects in the "Projects" section: create a new or add existing.

<u>Documentation is your best resource for using any libraries you import</u>

pls pls pls read documentation

platformio.ini



```ini
; PlatformIO Project Configuration File
;
;   Build options: build flags, source filter
;   Upload options: custom upload port, speed and extra flags
;   Library options: dependencies, extra library storages
;   Advanced options: extra scripting
;
; Please visit documentation for the other options and examples
; https://docs.platformio.org/page/projectconf.html

[env:upesy_wroom]
platform = espressif32 ; auto
board = upesy_wroom ; auto
framework = arduino ; auto
lib_deps = madhephaestus/ESP32Servo@^3.0.8 ; auto
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/sg90_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?

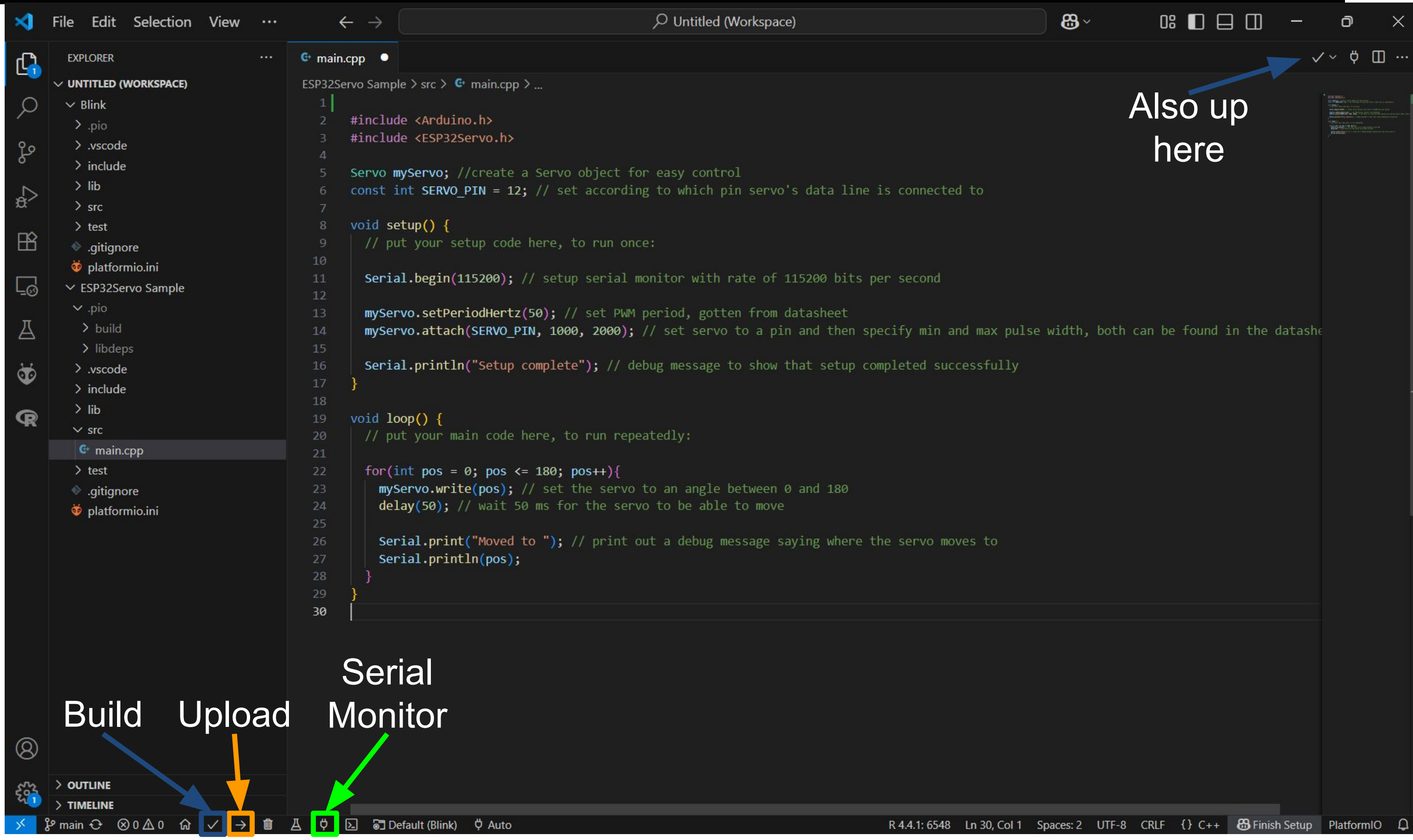
<u>YES!!!!</u>

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/HCSR04.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



```cpp

#include <Arduino.h>
#include <ESP32Servo.h>

Servo myServo; //create a Servo object for easy control
const int SERVO_PIN = 12; // set according to which pin servo's data line is connected to

void setup() {
  // put your setup code here, to run once:

  Serial.begin(115200); // setup serial monitor with rate of 115200 bits per second

  myServo.setPeriodHertz(50); // set PWM period, gotten from datasheet
  myServo.attach(SERVO_PIN, 1000, 2000); // set servo to a pin and then specify min and max pulse width, both can be found in the datashe

  Serial.println("Setup complete"); // debug message to show that setup completed successfully
}

void loop() {
  // put your main code here, to run repeatedly:

  for(int pos = 0; pos <= 180; pos++){
    myServo.write(pos); // set the servo to an angle between 0 and 180
    delay(50); // wait 50 ms for the servo to be able to move

    Serial.print("Moved to "); // print out a debug message saying where the servo moves to
    Serial.println(pos);
  }
}
```

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