RPI, GIT, Github, WiKi

Introduction to Single Board Computer (SBC)

An overview of Raspberry Pi zero 2w

Overview

Introduction to SBC

Overview of Raspberry Pi Zero 2 W and its benefits

Getting started with RPi Zero 2 W

Programming and development with RPi Zero 2W

What is a Single Board Computer (SBC)?

A complete computer built on a single circuit board.

Processor, memory, storage, and input/output options, all in one compact package.

Affordable compared to traditional PCs

Can be used for a variety of projects, from education to industrial applications

Low power consumption

Popular SBCs

- Raspberry Pi
- Orange Pi
- BeagleBone

Fig: Raspberry Pi

Fig: Orange Pi

Fig: BegaleBone Black

Raspberry Pi Zero 2W: An Introduction

A small, low-cost, and powerful version of the Raspberry Pi

Ideal for embedded AI projects

Quad-core ARM Cortex-A53 (1 GHz), 512 MB LPDDR2, Wi-Fi (802.11 b/g/n), Bluetooth 4.2

Mini HDMI for display and microSD card for OS and data

40 GPIO pins

Requires 5V power supply via Micro-USB

Smaller than a credit card (65mm x 30mm)

Why Raspberry Pi Zero 2W?

- Very affordable (~$15) and low power consumption
- Small enough for portable or embedded applications
- Good online community support for troubleshooting, tutorials, and projects
- Ideal for learning programming, electronics and prototyping

Fig: Raspberry Pi Zero 2 W

Getting Started with RPi Zero 2W

- Download Raspberry Pi OS from the official website
- Flash the OS onto the microSD card using tools like Etcher
- Insert the microSD card into the Raspberry Pi Zero 2W
- Power up and follow on-screen instructions to complete the setup

Programming and Development on RPi Zero

- Python is most common for embedded ML projects
- C/C++ can be used or more complex projects with hardware control
- A Linux-based OS (Raspberry Pi OS) designed specifically for the Raspberry Pi
- Can be run in GUI or headless mode (without a monitor)
- GPIO libraries for interfacing with external sensors, LEDs, motors
- PiCamera for camera-based projects

Anaconda Software

An introduction and basic overview of the features

Overview

- Introduction to Anaconda
- Advantages and limitations
- Key components of Anaconda
- Virtual environment in Anaconda

What is Anaconda?

- A free and open-source software distribution
- Designed for Python and R programming
- Simplifies package management and deployment
- Widely used in Data Science, Machine Learning, Scientific Computing

Advantages of Anaconda

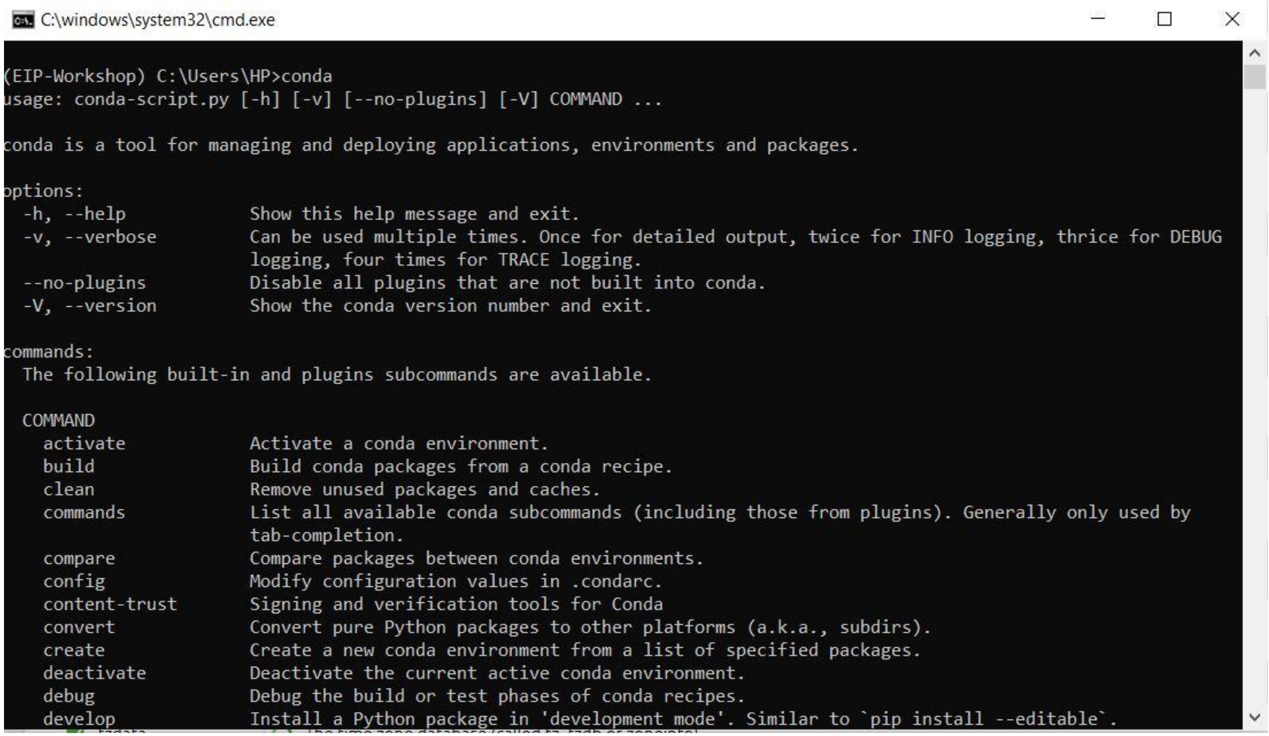
- Easy to install and use
- All-in-one platform
- Free for educational use
- Strong community support
- Saves time for beginners

Limitations of Anaconda

- Large installation size
- May use more disk space
- Not always needed for very small projects

Key Components of Anaconda

- Python Interpreter
  - Conda – Package and environment manager
  - Anaconda Navigator – Graphical User Interface (GUI)
- Popular development tools:
  - Jupyter Notebook
  - Spyder IDE



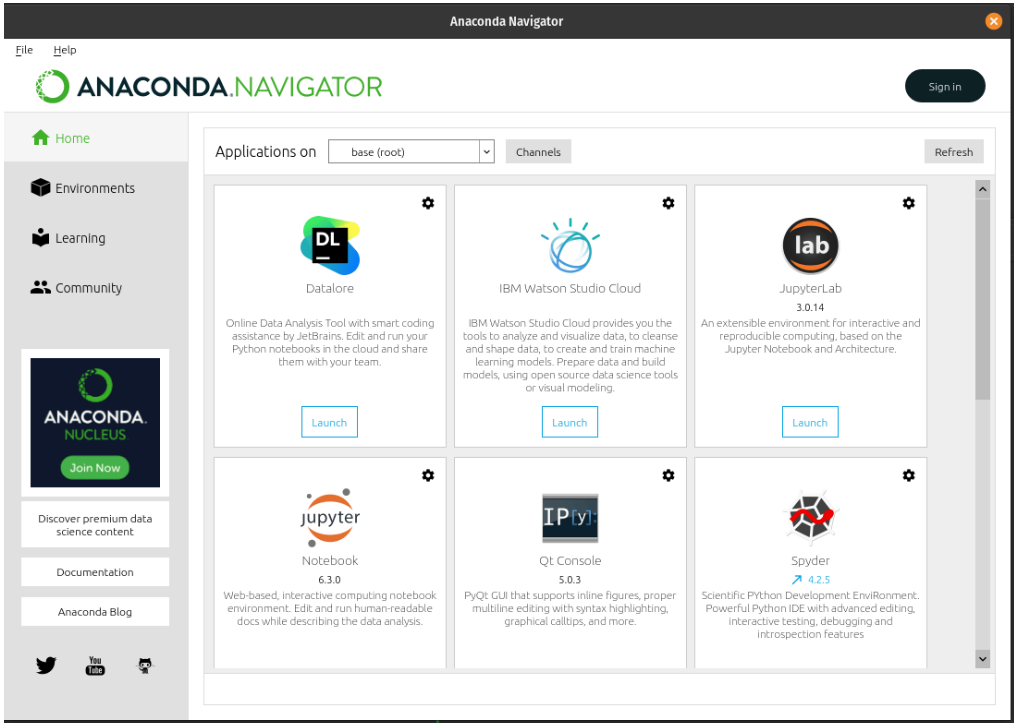
```
(EIP-Workshop) C:\Users\HP>conda
usage: conda-script.py [-h] [-v] [--no-plugins] [-V] COMMAND ...

conda is a tool for managing and deploying applications, environments and packages.

options:
  -h, --help            Show this help message and exit.
  -v, --verbose         Can be used multiple times. Once for detailed output, twice for INFO logging, thrice for DEBUG
                        logging, four times for TRACE logging.
  --no-plugins          Disable all plugins that are not built into conda.
  -V, --version         Show the conda version number and exit.

commands:
  The following built-in and plugins subcommands are available.

  COMMAND
    activate            Activate a conda environment.
    build               Build conda packages from a conda recipe.
    clean               Remove unused packages and caches.
    commands            List all available conda subcommands (including those from plugins). Generally only used by
                        tab-completion.
    compare             Compare packages between conda environments.
    config              Modify configuration values in .condarc.
    content-trust       Signing and verification tools for Conda
    convert             Convert pure Python packages to other platforms (a.k.a., subdirs).
    create              Create a new conda environment from a list of specified packages.
    deactivate          Deactivate the current active conda environment.
    debug               Debug the build or test phases of conda recipes.
    develop             Install a Python package in 'development mode'. Similar to `pip install --editable`.
```

Fig: Conda Terminal

Anaconda Navigator

- GUI-based application launcher
- No command-line knowledge required
- Used to:
  - Launch Jupyter Notebook
  - Launch Spyder
  - Manage environments and packages
  - Beginner-friendly interface

Fig: Anaconda Navigator



Datalore

Online Data Analysis Tool with smart coding assistance by JetBrains. Edit and run your Python notebooks in the cloud and share them with your team.

IBM Watson Studio Cloud

IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.

JupyterLab
3.0.14

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
6.3.0

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Qt Console
5.0.3

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

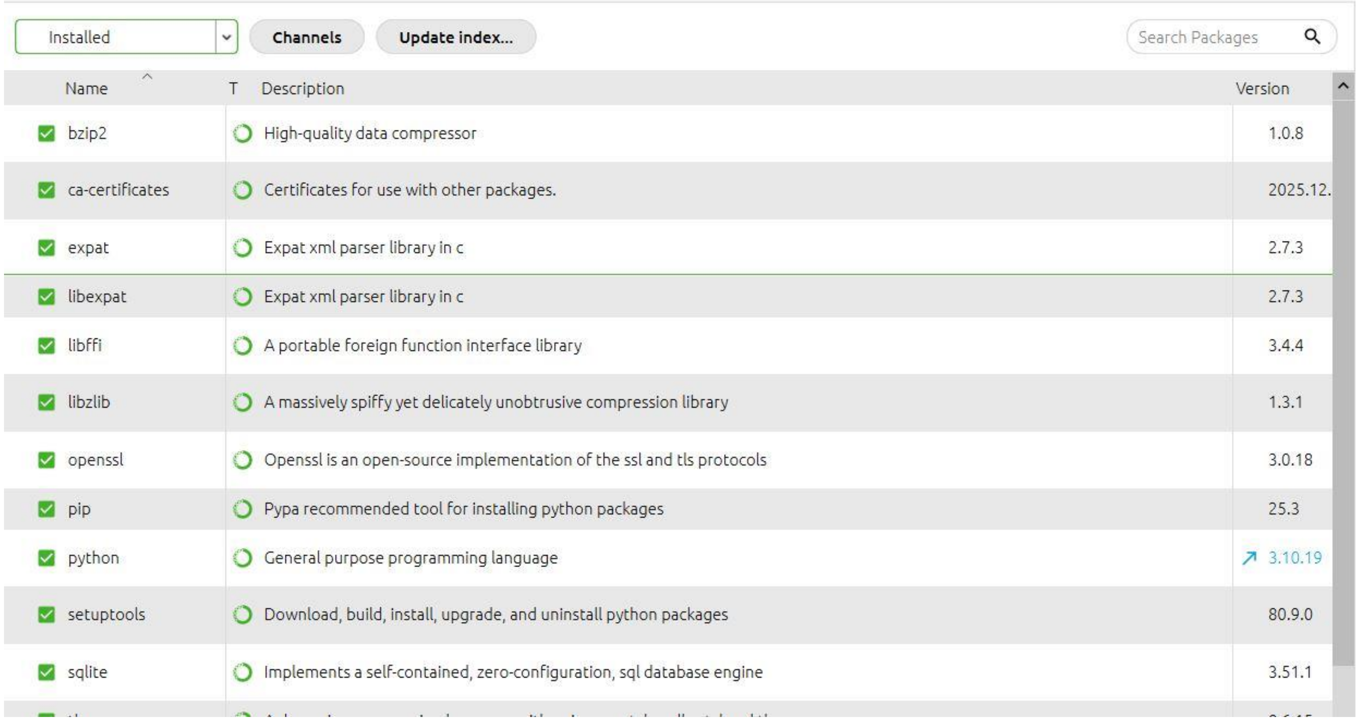
Spyder
↗ 4.2.5

Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Conda Package Manager

- Manages:
  - Libraries (NumPy, Pandas, etc.)
  - Software dependencies
- Advantages:
  - Avoids version conflicts
  - Cross platform
  - Simple to install/update packages

Fig: Package management

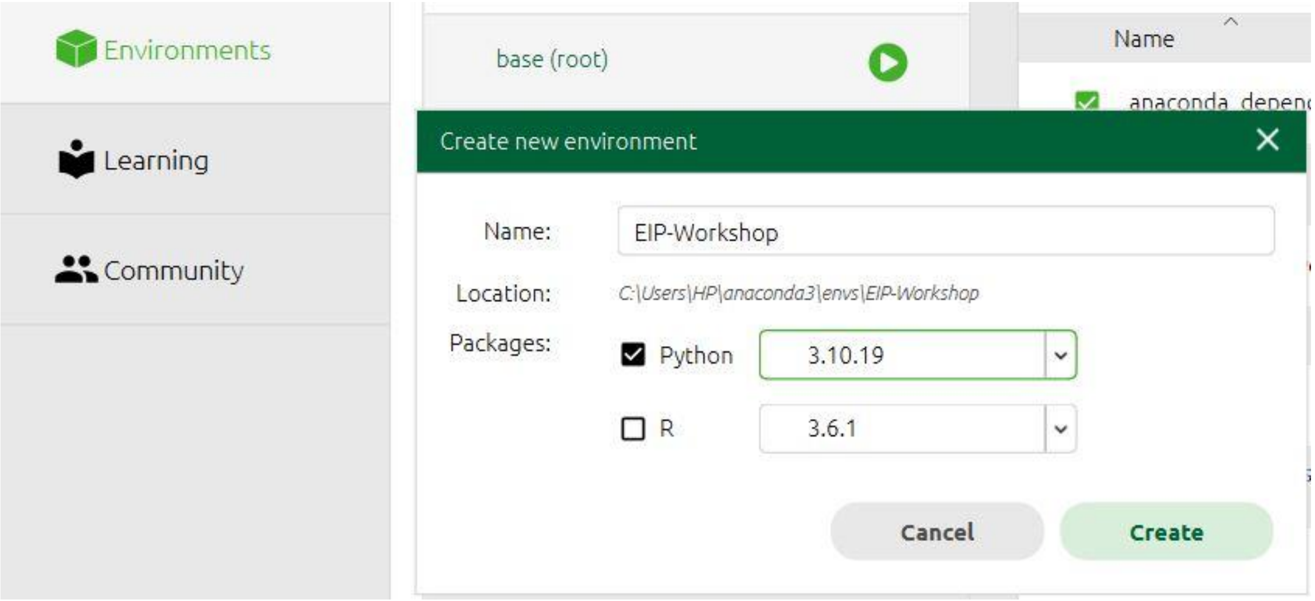


	Name	T	Description	Version
☑	bzip2	○	High-quality data compressor	1.0.8
☑	ca-certificates	○	Certificates for use with other packages.	2025.12.
☑	expat	○	Expat xml parser library in c	2.7.3
☑	libexpat	○	Expat xml parser library in c	2.7.3
☑	libffi	○	A portable foreign function interface library	3.4.4
☑	libzlib	○	A massively spiffy yet delicately unobtrusive compression library	1.3.1
☑	openssl	○	Openssl is an open-source implementation of the ssl and tls protocols	3.0.18
☑	pip	○	Pypa recommended tool for installing python packages	25.3
☑	python	○	General purpose programming language	↗ 3.10.19
☑	setuptools	○	Download, build, install, upgrade, and uninstall python packages	80.9.0
☑	sqlite	○	Implements a self-contained, zero-configuration, sql database engine	3.51.1

Virtual Environments in Anaconda

- Virtual environment = isolated workspace
- Allows:
  - Different Python versions
  - Different library versions
- Useful for:
  - Multiple projects
  - Team collaboration
- Prevents software conflicts

Fig: Virtual environment creation



Name: EIP-Workshop

Location: C:\Users\HP\anaconda3\envs\EIP-Workshop

Packages:

☑ Python 3.10.19

□ R 3.6.1

Introduction to Git

A simple overview

Overview

- Version Control System (VCS)
- Git as a VCS
- Basic Git terminology and workflow
- Common Git commands

Version Control System (VCS)

- Keeps history of changes
- Helps avoid losing work
- Makes collaboration easier
- Allows rollback to previous versions
- Used for code, documents, designs, and reports

What is Git?

- Git is a version control system (a software)
- Works locally on your computer
- Tracks changes in files over time
- Created by Linus Torvalds
- Very fast and widely used

Fig: Git logo

Basic Git Terminology

- *Repository (repo)* – project folder tracked by Git
- *Commit* – snapshot of changes
- *Branch* – parallel version of code
- *Main / Master* – default branch
- *Clone* – copy a repo
- *Push / Pull* – send or receive changes

Basic Git Workflow

- Edit files
- Stage changes
- Commit changes
- Push to remote server (for example GitHub)
- Pull updates from others

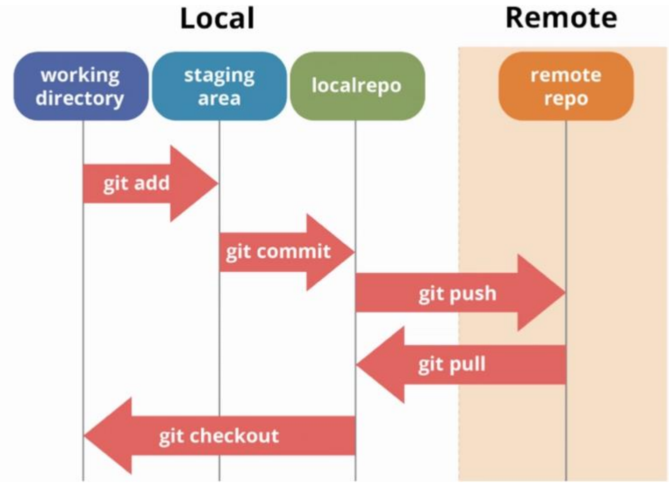


Fig: Simple git workflow

Common Git Commands

- *git init* – start a repository
- *git status* – check status
- *git add* – stage changes
- *git commit* – save changes
- *git push* – upload changes
- *git pull* – download changes

Branching Concept

- Branch = independent line of development
- Main branch stays stable
- New features developed in separate branches
- Multiple people can work simultaneously

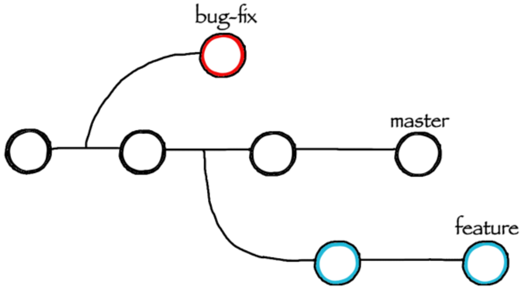


Fig: git branch concept

Introduction to GitHub

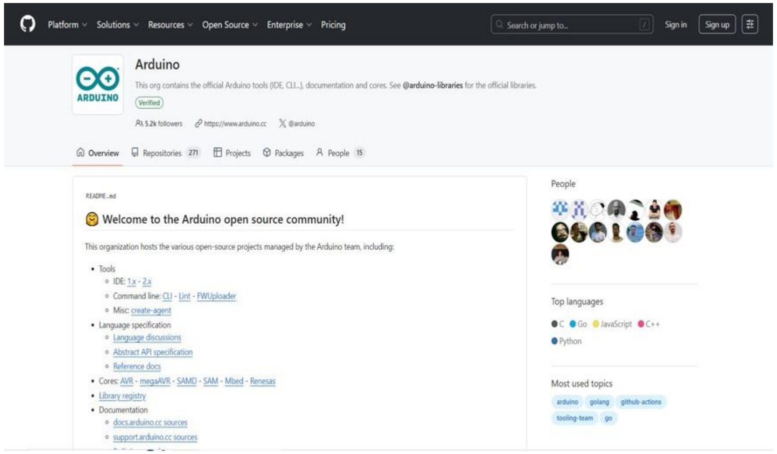
A basic Overview

Overview

- Introduction to GitHub
- Usage of GitHub
- GitHub Collaboration features
- Best practices to use GitHub
- Documentation on GitHub
- Benefits for students using GitHub

What is GitHub?

- GitHub is a cloud-based hosting service for Git
- Stores Git repositories online
- Enables collaboration and sharing
- Provides backup and access from anywhere
- Popular in open-source and industry



Arduino

This org contains the official Arduino tools (IDE, CLI...), documentation and cores. See **@arduino-libraries** for the official libraries.

5.2k followers https://www.arduino.cc @arduino

README.md

Welcome to the Arduino open source community!

This organization hosts the various open-source projects managed by the Arduino team, including:

- Tools
  - IDE: 1.x - 2.x
  - Command line: CLI - Lint - FWUploader
  - Misc: create-agent
- Language specification
  - Language discussions
  - Abstract API specification
  - Reference docs
- Cores: AVR - megaAVR - SAMD - SAM - Mbed - Renesas
- Library registry
- Documentation
  - docs.arduino.cc sources
  - support.arduino.cc sources

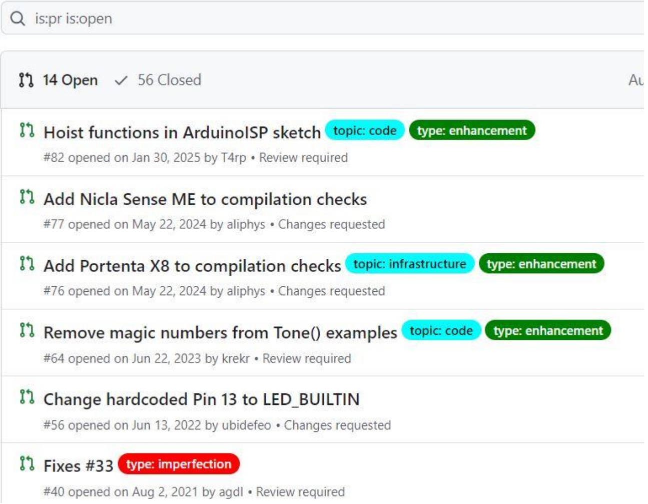
Fig: Arduino GitHub Repository

When to Use GitHub?

- Team projects
- Open-source contributions
- Project backup
- Code review
- Resume and portfolio building

GitHub Collaboration Features

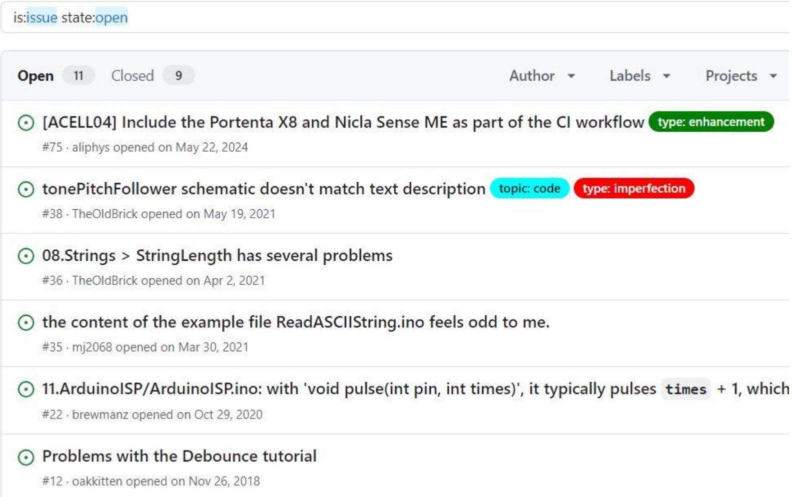
- Pull Requests – propose changes
- Issues – track bugs and tasks
- Forks – personal copy of a project
- Actions – automation and CI/CD



is:pr is:open

14 Open ✓ 56 Closed Au

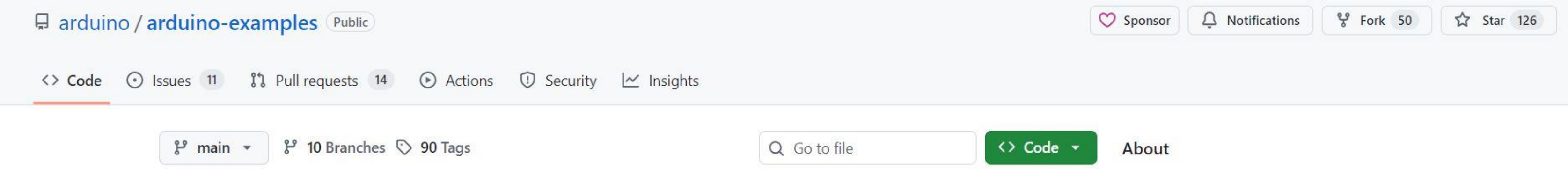
- Hoist functions in ArduinoISP sketch topic: code type: enhancement
  #82 opened on Jan 30, 2025 by T4rp • Review required
- Add Nicla Sense ME to compilation checks
  #77 opened on May 22, 2024 by aliphys • Changes requested
- Add Portenta X8 to compilation checks topic: infrastructure type: enhancement
  #76 opened on May 22, 2024 by aliphys • Changes requested
- Remove magic numbers from Tone() examples topic: code type: enhancement
  #64 opened on Jun 22, 2023 by krekr • Review required
- Change hardcoded Pin 13 to LED_BUILTIN
  #56 opened on Jun 13, 2022 by ubidefeo • Changes requested
- Fixes #33 type: imperfection
  #40 opened on Aug 2, 2021 by agdl • Review required



is:issue state:open

Open 11 Closed 9 Author ▾ Labels ▾ Projects ▾

- [ACELL04] Include the Portenta X8 and Nicla Sense ME as part of the CI workflow type: enhancement
  #75 · aliphys opened on May 22, 2024
- tonePitchFollower schematic doesn't match text description topic: code type: imperfection
  #38 · TheOldBrick opened on May 19, 2021
- 08.Strings > StringLength has several problems
  #36 · TheOldBrick opened on Apr 2, 2021
- the content of the example file ReadASCIIString.ino feels odd to me.
  #35 · mj2068 opened on Mar 30, 2021
- 11.ArduinoISP/ArduinoISP.ino: with 'void pulse(int pin, int times)', it typically pulses `times` + 1, which
  #22 · brewmanz opened on Oct 29, 2020
- Problems with the Debounce tutorial
  #12 · oakkitten opened on Nov 26, 2018



arduino / arduino-examples Public

Sponsor | Notifications | Fork 50 | Star 126

Code | Issues 11 | Pull requests 14 | Actions | Security | Insights

main ▾ | 10 Branches | 90 Tags | Go to file | Code ▾ | About

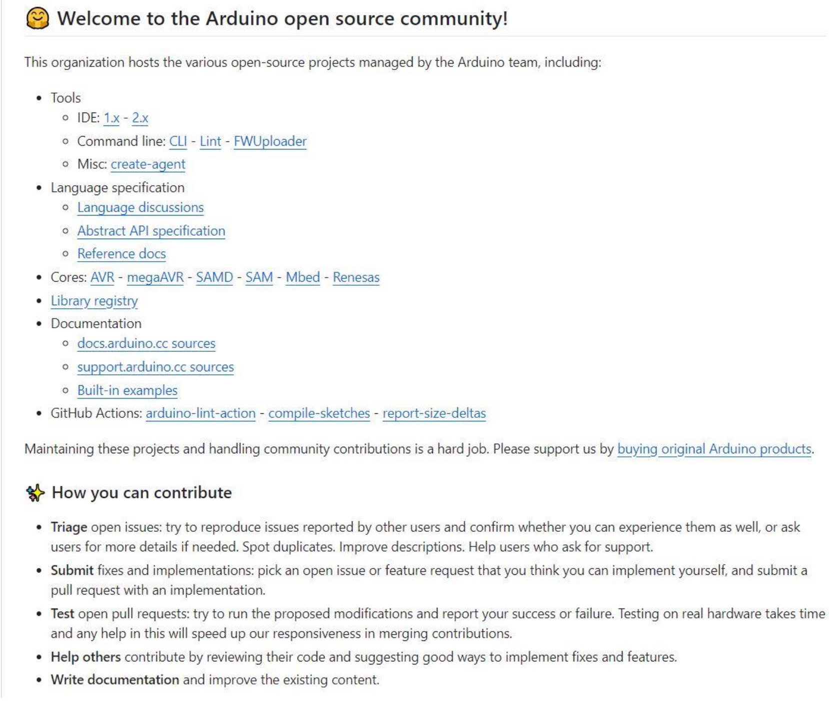
Fig: GitHub collaboration Features

GitHub Best Practices

- Commit frequently
- Write clear commit messages
- Pull before pushing
- Use branches for features
- Keep repositories organized

Creating Documentation Using GitHub

- Commonly used to host project documentation
- Documentation lives inside the repository
- Written in simple text formats (Markdown)
- Easy to update and collaborate on



🤗 Welcome to the Arduino open source community!

This organization hosts the various open-source projects managed by the Arduino team, including:

- Tools
  - IDE: 1.x - 2.x
  - Command line: CLI - Lint - FWUploader
  - Misc: create-agent
- Language specification
  - Language discussions
  - Abstract API specification
  - Reference docs
- Cores: AVR - megaAVR - SAMD - SAM - Mbed - Renesas
- Library registry
- Documentation
  - docs.arduino.cc sources
  - support.arduino.cc sources
  - Built-in examples
- GitHub Actions: arduino-lint-action - compile-sketches - report-size-deltas

Maintaining these projects and handling community contributions is a hard job. Please support us by buying original Arduino products.

✨ How you can contribute

- **Triage** open issues: try to reproduce issues reported by other users and confirm whether you can experience them as well, or ask users for more details if needed. Spot duplicates. Improve descriptions. Help users who ask for support.
- **Submit** fixes and implementations: pick an open issue or feature request that you think you can implement yourself, and submit a pull request with an implementation.
- **Test** open pull requests: try to run the proposed modifications and report your success or failure. Testing on real hardware takes time and any help in this will speed up our responsiveness in merging contributions.
- **Help others** contribute by reviewing their code and suggesting good ways to implement fixes and features.
- **Write documentation** and improve the existing content.

Fig: Documentation in GitHub

What is Markdown?

- Lightweight formatting language
- Easy to read and write
- Uses simple symbols (#, *, -)
- Automatically rendered by GitHub

Common Documentation Files on GitHub

- README.md
  - Main project description
  - Explains what the project does
  - Shows how to install and use it
- CONTRIBUTING.md
  - Rules for contributing
- LICENSE
  - Legal usage terms
- docs/ folder
  - Detailed documentation files

GitHub Pages (Documentation Websites)

- GitHub can host documentation as a website
- Uses Markdown + static site generators
- Common tools: Jekyll, MkDocs
- Free hosting for public repositories

GitHub Benefits for Students

- Industry-relevant skill
- Teamwork experience
- Organized project history
- Easy project submission
- Public portfolio for internships/jobs