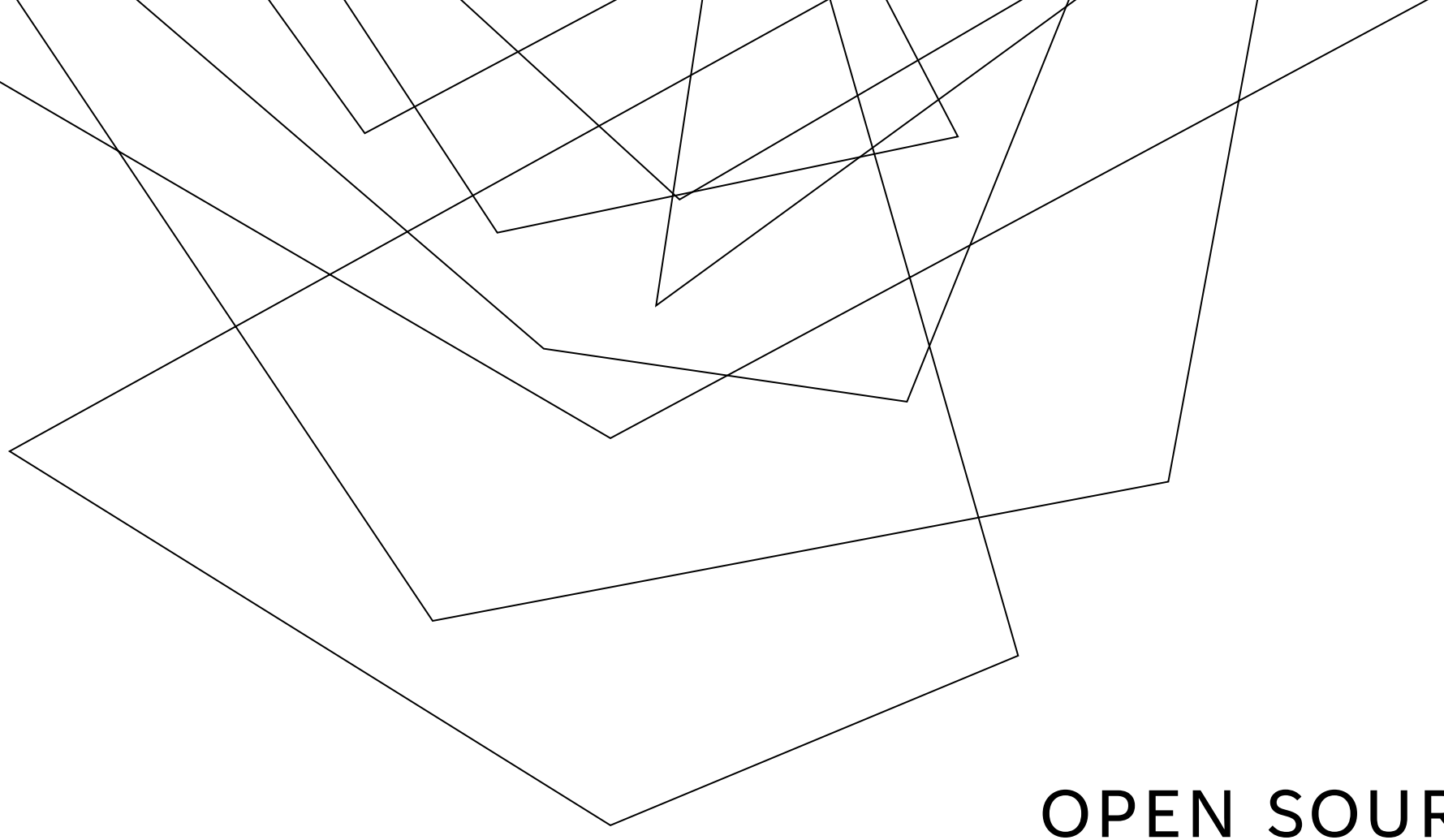


OPEN SOURCE HARDWARE

Manipal Open Source Society (MOSS)

AGENDA

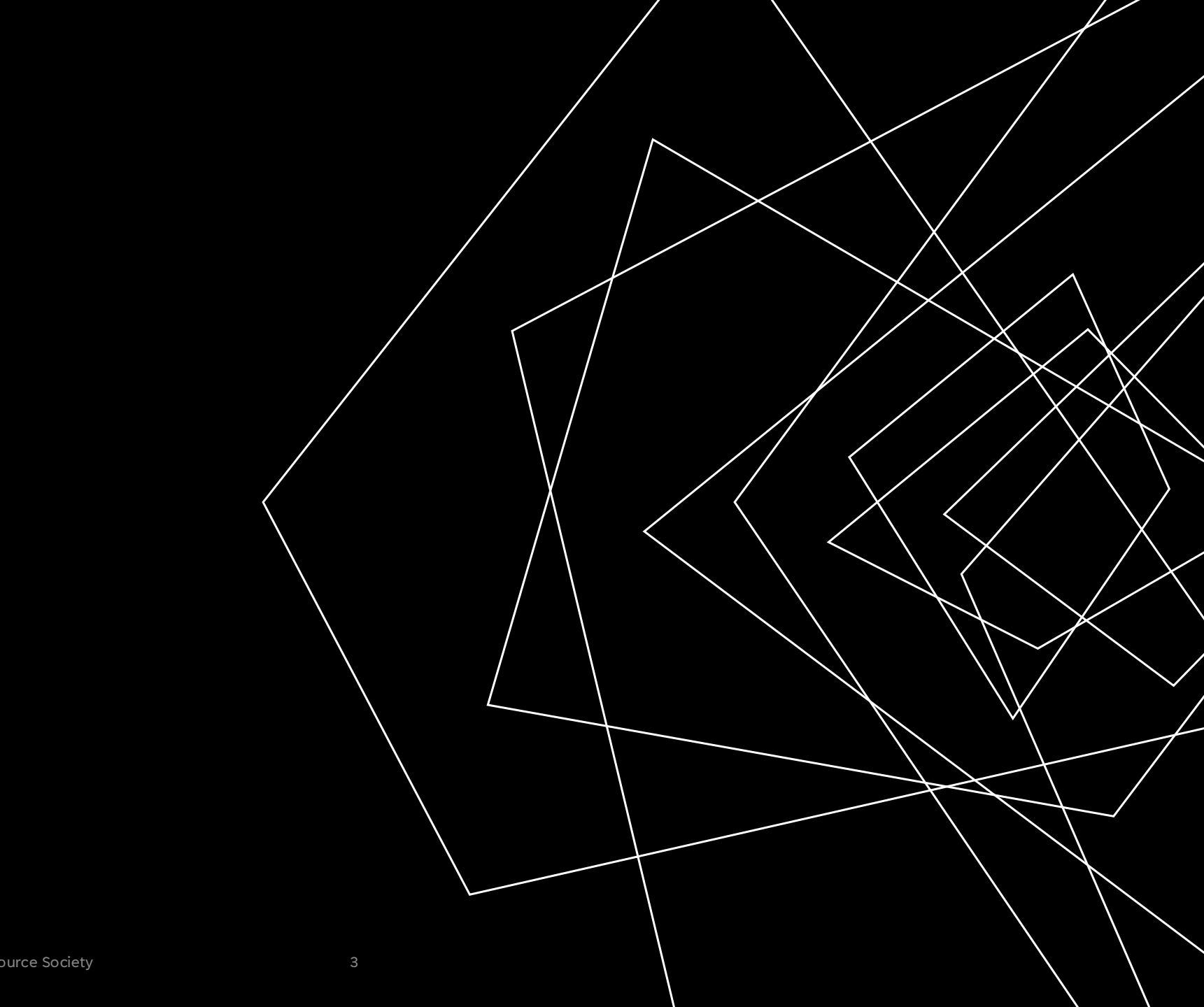
Power

Impact

Contribute

Learn

Next Steps



A LITTLE ABOUT ME

Been working with the Raspberry Pi, STM32 and Nvidia Jetson ecosystems for over 5 years now, having built a custom kernel for the Cortex A-53 and merged contributions to the Arduino library source code.

I've also designed and built 5+ robotic systems and several PCBs for prototypes including GPIO-enabled voice assistant, electric R/C skateboard, PianoTiles bot etc

PS: I also built a 4Hz redstone computer in Minecraft :)

OPEN SOURCE

Documenting and sharing ideas has always driven innovation.

It can be through scientific research, reviewed by peers in journals- giving rise to international collaboration between people who otherwise would never be able to exchange ideas, or in a slightly different environment,

Through file hosting and version control platforms like GitHub and SourceForge – where engineers from all walks of life come together to work on someone else's years' worth of work and take it forward.

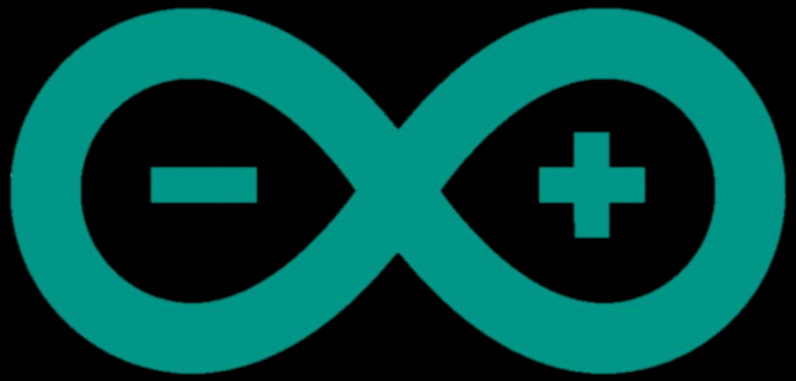
EVOLUTION OF THE BICYCLE

1818

1860

1885

2018

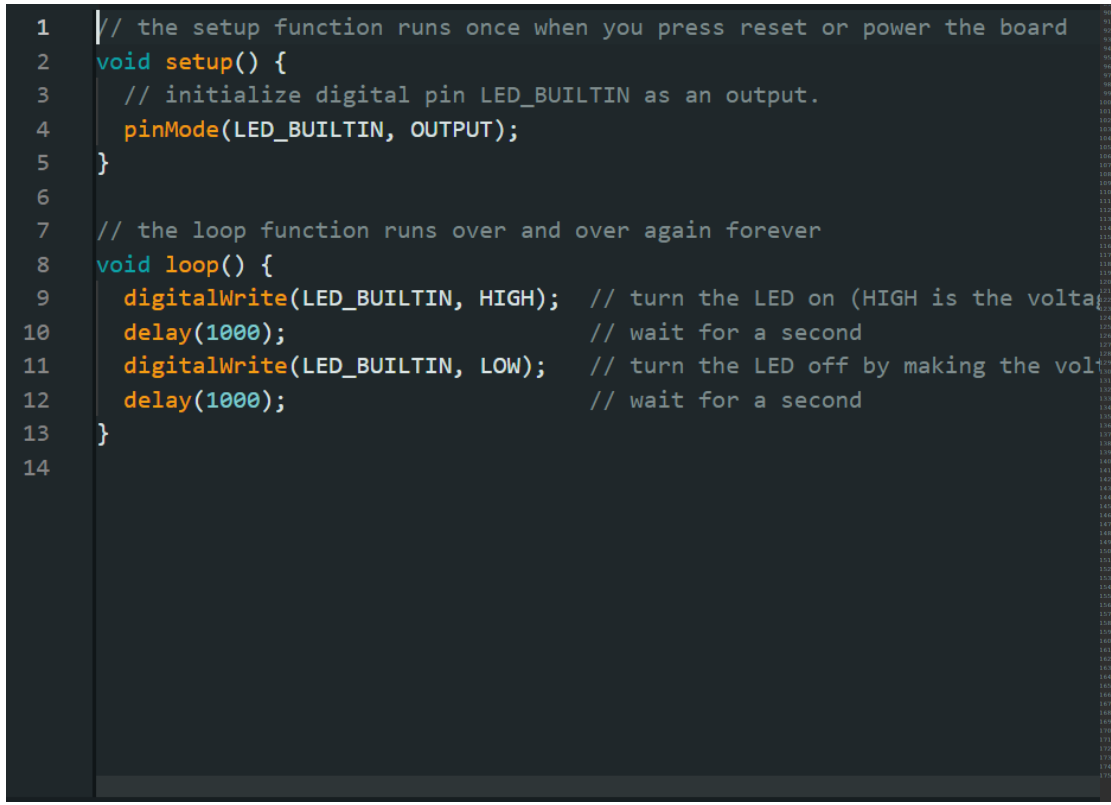


ARDUINO

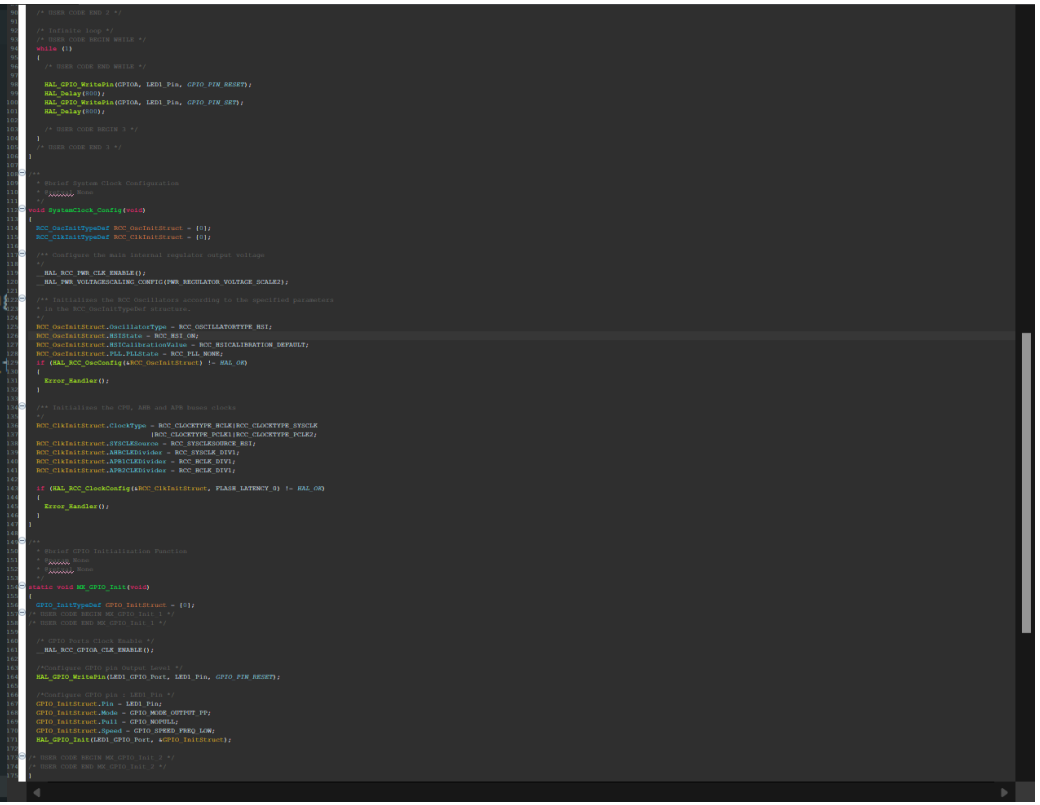
The world’s most popular open source electronics prototyping platform, with their microcontroller development boards literally used by kids and industries alike.

It also hosts and maintains heavily abstracted software libraries which make it easy to interface several complex sensors, actuators and protocols using their Arduino IDE.

BLINKING AN LED WITH ARDUINO vs STM32



```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the volta
  delay(1000);                      // wait for a second
  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the vol
  delay(1000);                      // wait for a second
}
```



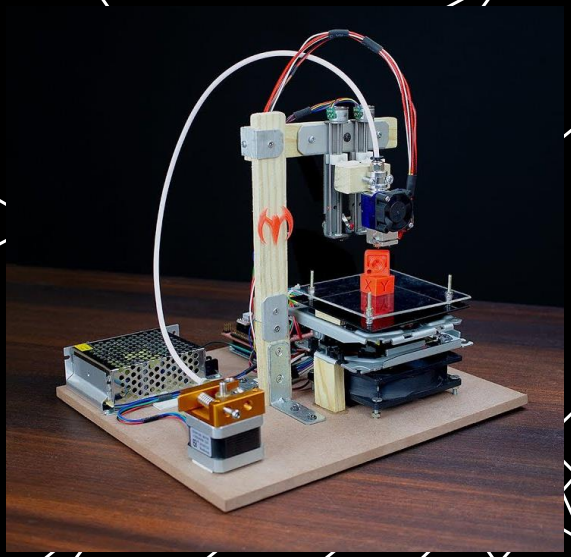
Arduino IDE

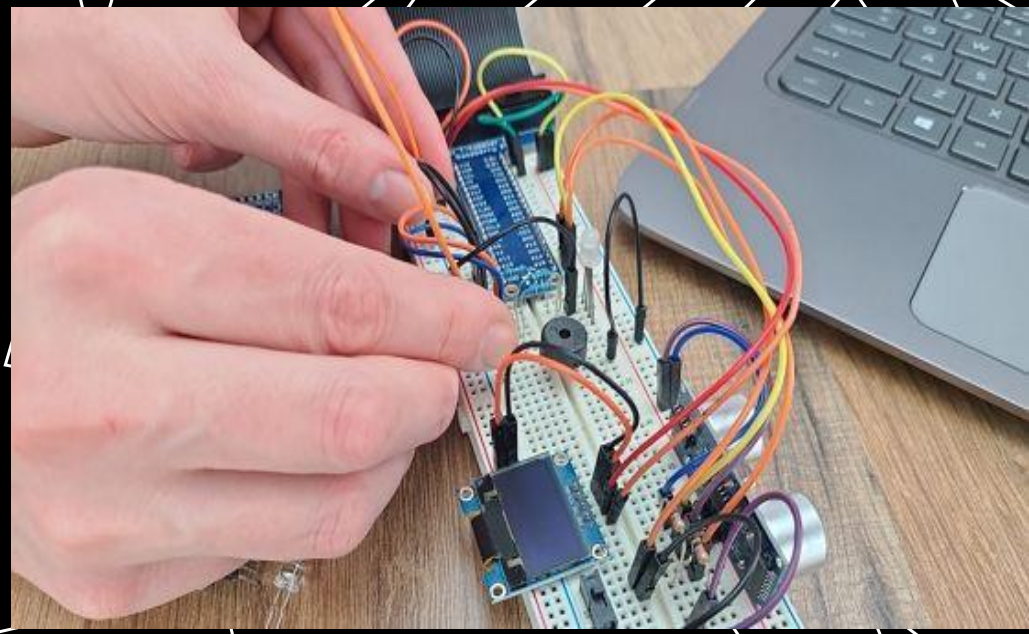
STM32CubeIDE

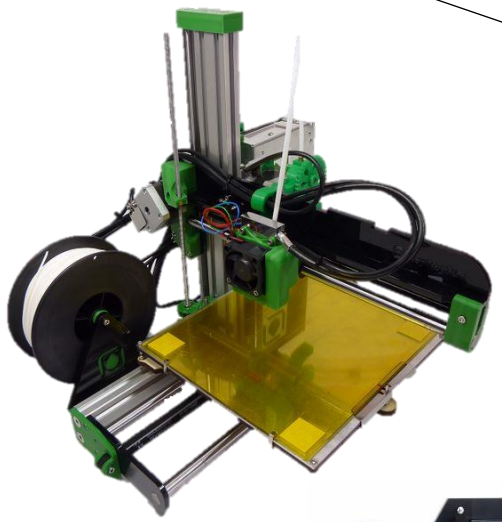
THE ARDUINO COMMUNITY

Clockwise, from top left:

An Arduino Uno R3, DIY 3D Printer, IoT Intruder Alarm Prototype, Arduino User Group Germany, DIY Electric Skateboard built with an Arduino Mega





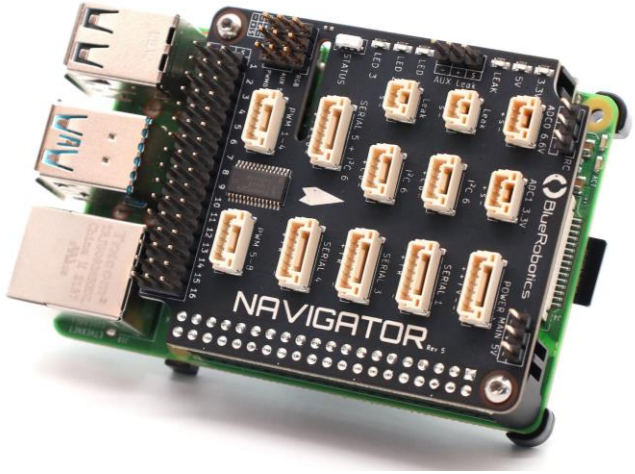


RepRap

Open designs for low-cost accessible self-replicating 3D printer, modified by the community to print PCBs as well as recycle polyethylene to produce printer filament.

Essentially allows you to print and assemble a 3D printer on your own from a local maker lab, if you can’t afford to pay 60k for a proprietary one.

Comprehensive hardware bundled with supporting ground control software that can turn literally any type of locomotive system autonomous – land-based rovers, aerial copters, mini submarines, boats, planes etc.



THE ARDUPILOT COMMUNITY

Clockwise, from top left:

ArduPilot Navigator Controller, Autonomous Rover, Unmanned Airboat, ArduSub, VTOL SAR Aircraft on ArduPilot

STRUCTURE

What does OSH look like?

According to the OSHWA, an open-source hardware project should consist of:

- **Hardware** - The physical functional components/elements of the product incl. schematics, BoM, gerbers
- **Software** - Any code, firmware, or software involved in product's function
- **Documentation** - Design files, firmware docs, onboarding guides, etc.
- **Branding** - Brand names, product names, logos, and product designs (optional, but recommended)

Licensing

Open, non-restrictive licenses that support the freedom to use, study, modify, share and distribute hardware designs, and products based on those designs

Purpose-built open hardware license built by CERN for its hardware projects hosted on the Open Hardware Repository.

Several popular projects like the Arduino still operate on generic open-source licenses like Creative Commons (CC BY-SA) and the GNU GPL series

Files

master

Go to file t

- .github
- bootloaders
- cores/arduino
- drivers
- extras
- firmwares
- libraries
  - EEPROM
  - HID
  - SPI
  - SoftwareSerial
  - Wire
    - examples
    - src
    - keywords.txt
    - library.properties
- variants
- .codespellrc
- README.md
- boards.txt
- platform.txt

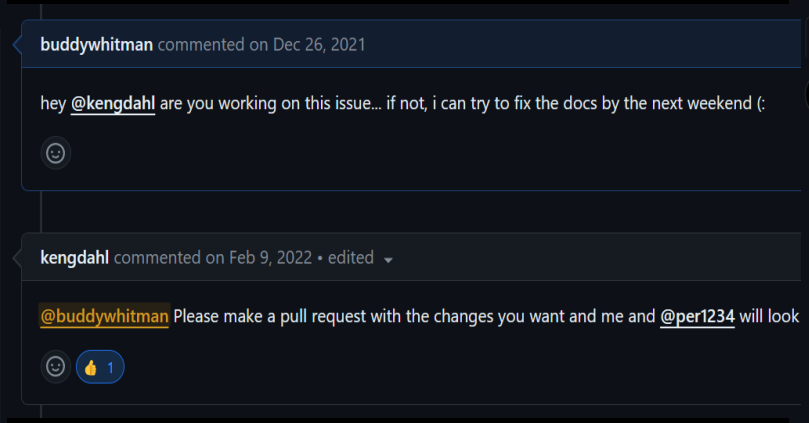
ArduinoCore-avr / cores / arduino /

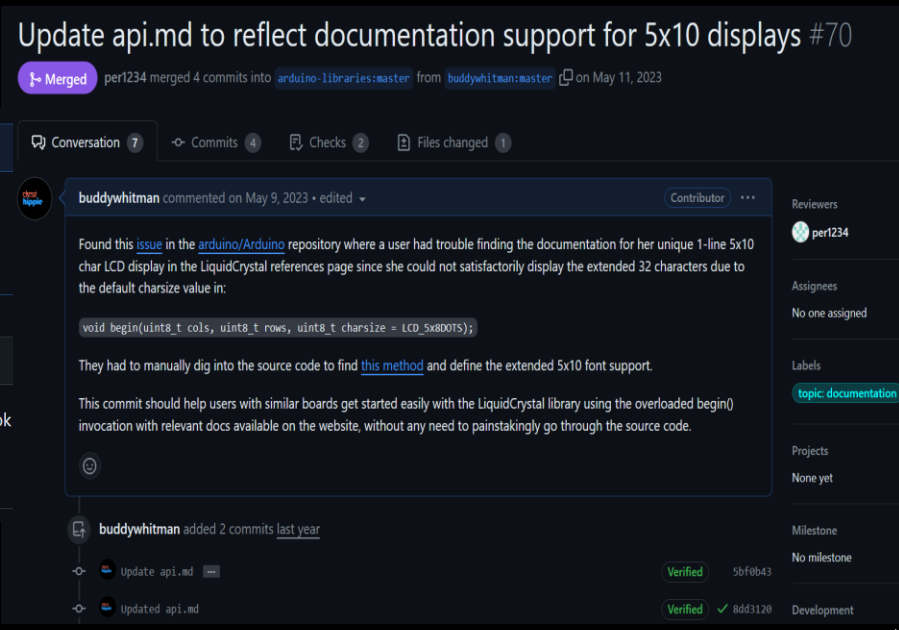
Add file

per1234 Fix misspelled words in comments 4d8d41e · 2 years ago History

Name	Last commit message	Last commit date
..		
Arduino.h	Merge branch 'master' into patch-1	5 years ago
CDC.cpp	Fix misspelled words in comments	2 years ago
Client.h	Added license for Client, IPAddressm and Server	11 years ago
HardwareSerial.cpp	Correct typos in comments and documentation	3 years ago
HardwareSerial.h	Correct typos in comments and documentation	3 years ago
HardwareSerial0.cpp	Fix sine -> since typo in HardwareSerial files	4 years ago
HardwareSerial1.cpp	Fix sine -> since typo in HardwareSerial files	4 years ago
HardwareSerial2.cpp	Fix sine -> since typo in HardwareSerial files	4 years ago
HardwareSerial3.cpp	Fix sine -> since typo in HardwareSerial files	4 years ago
HardwareSerial_private.h	Correct typos in comments and documentation	3 years ago
IPAddress.cpp	Remove old TODOs for non-standard ipv4 format support	8 years ago
IPAddress.h	Fixed another regression in IPAddress.h	9 years ago
PluggableUSB.cpp	[USB] use plugged modules name to create iSerial field	9 years ago
PluggableUSB.h	[USB] use plugged modules name to create iSerial field	9 years ago
Print.cpp	Change double quotes to single quotes	8 years ago

CONTRIBUTING TO THE ARDUINO DOCS





1. Search

For the good first issue filter in a repository that looks interesting

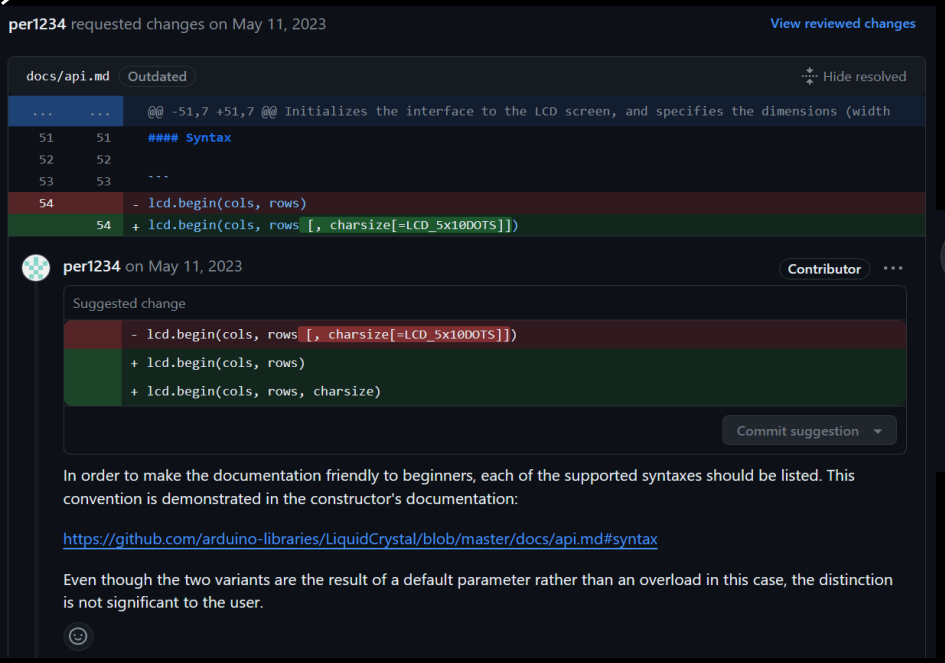
2. Communicate

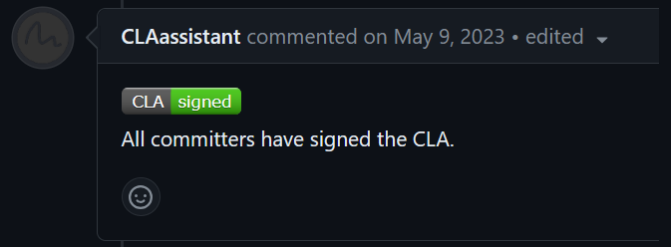
Your intention to work on the issue, resolve conflicts

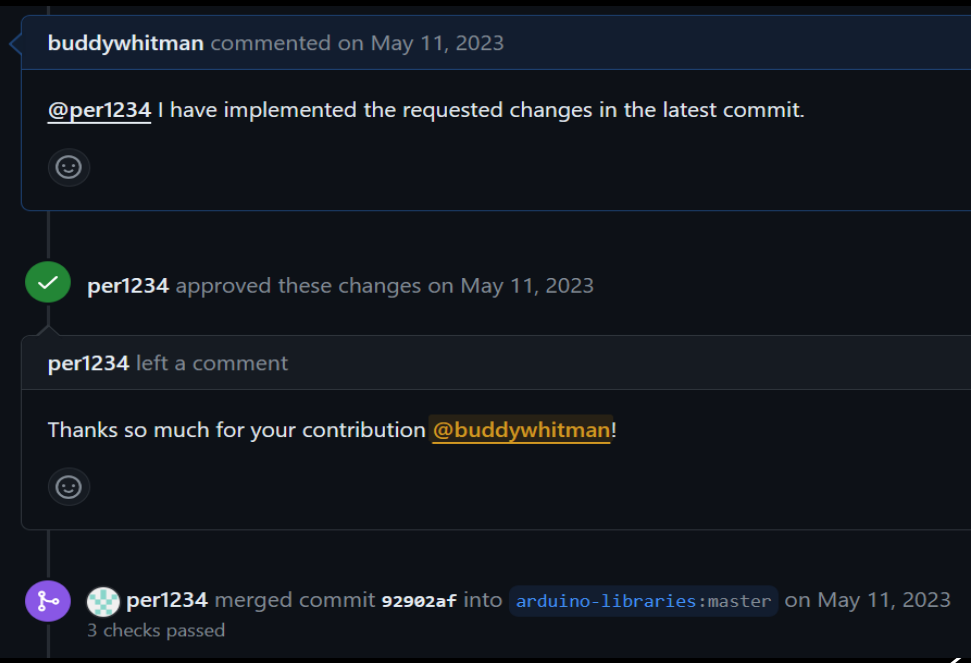
3. Figure

Out the exact file to be modified, the repository, conventions, tech stack

CONTRIBUTING TO THE ARDUINO DOCS 2







4. Iterate

On the changes suggested in code review, write clean code and test

5. Sign

The Contributor License Agreement, right before creating a pull request

6. Merge

Assure the maintainer using screenshots, metrics that the changes have been met

WHERE CAN I START?

embedded systems	hardware IC design	communication/ RF networks	graphics/ shaders APIs	mechatronics
Develop computers embedded in devices (consumer IoT appliances, automotive aerospace electronics, OS-level firmware).	Creating intricate digital/analog circuits (e.g., CPUs, memory chips) or application-specific ICs and SoCs at a microscopic scale.	Wireless communication technologies (e.g., Wi-Fi, cellular networks) and radio frequency/ microwave design.	Software interfaces for rendering graphics (e.g., games, 3D modelling) on screens– used by GPU makers like Nvidia & AMD	Build autonomous and robotic systems that power self-driving cars, military drones and additive manufacturing tech

HOW WE'LL GET THERE

Rigorous C/C++ Workflow

Learn the nuances of C/C++ including bitwise operators, memory access, graphics, header files and preprocessors

Learn cross-compilation tools and command line workflows like GCC

Getting Started with Arduino

Interface a variety of sensors, actuators and communication protocols using the Arduino breakout boards

Start with heavily abstracted libraries today, write bare-metal drivers tomorrow

Advanced Hardware Engineering

Hardware Design: design IP like CPUs, memory, other custom ICs and SoCs while working with FPGAs on Vivado, Virtuoso, implementing digital systems

Embedded Systems: develop baremetal firmware and breakout boards for the above designed components and build kernels, drivers for these devices using relevant assembly & COA

ADVANCED LL/ EMBEDDED SYSTEMS

Computer Architecture & Kernel Dev:

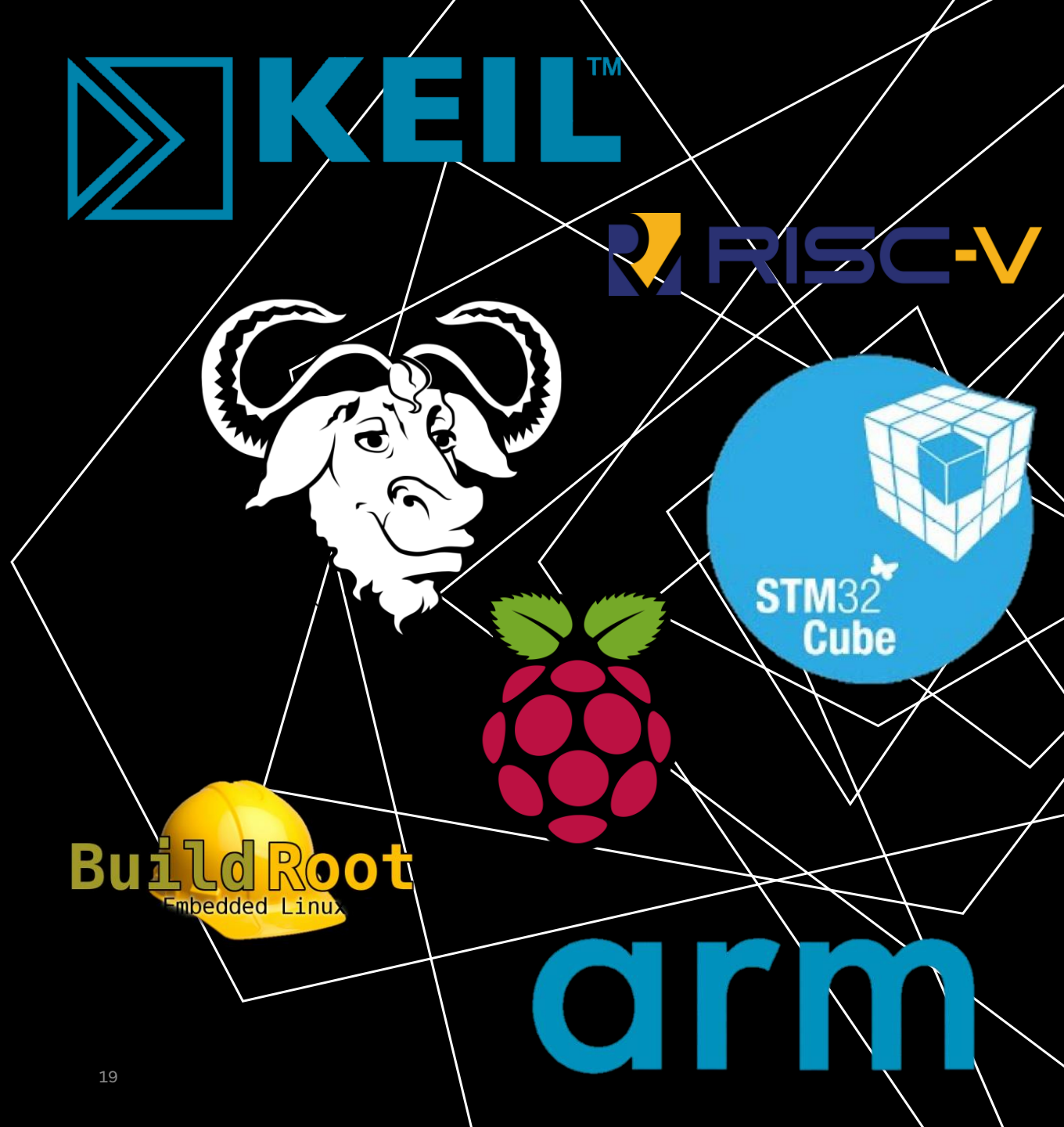
• master COA by learning the instruction sets of arm, risc-v or x86 and their execution

• work with the GNU embedded toolchain and tools like openocd, qemu, gdb, zephyr

Drivers, Firmware & Baremetal Dev:

• get started with an STM32 blue pill with the STM32CubeIDE or Keil uVision to write firmware for sensors, actuators and displays

• learn building embedded linux operating systems for higher-end devices using yocto and buildroot



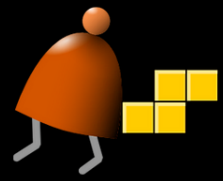
FPGA vs SoC vs ASIC

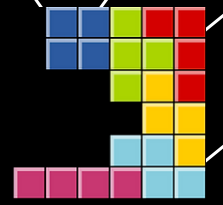
Field Programmable Gate Arrays (FPGAs)	Application-specific Integrated Circuits (ASICs)	System-on-Chips (SoCs)
Configurable integrated circuit made up of an array of programmable blocks linked by programmable interconnect. Can be reprogrammed after manufacturing to be reused for multiple digital system designs	Any custom-made chip with digital and/or analog functions with optimized performance and power consumption Incredibly expensive and resource-intensive to develop, FPGAs used for simulation in development process	A silicon chip that contains one or more processor cores and/or DSPs - with on-chip memory, peripheral functions, hardware accelerators Can be thought of as an ASIC with one/more microcontrollers and/or microprocessors

HARDWARE DESIGN

From Nand to Tetris

Building a Modern Computer From First Principles





Digital Logic Design:

- nand2tetris on coursera for an introduction
- digital system design coursework in college
- download Xilinx Vivado to build on FPGAs and SoCs, Cadence Virtuoso, Innovus for custom ICs
- learn the design workflow with tools like yosys, testbench, icestorm, nextpnr, gtkwave

PCB Design:

- refer Phil's Lab on YouTube for tutorials
- download and start with the KiCad series for smoother transition to Altium Designer

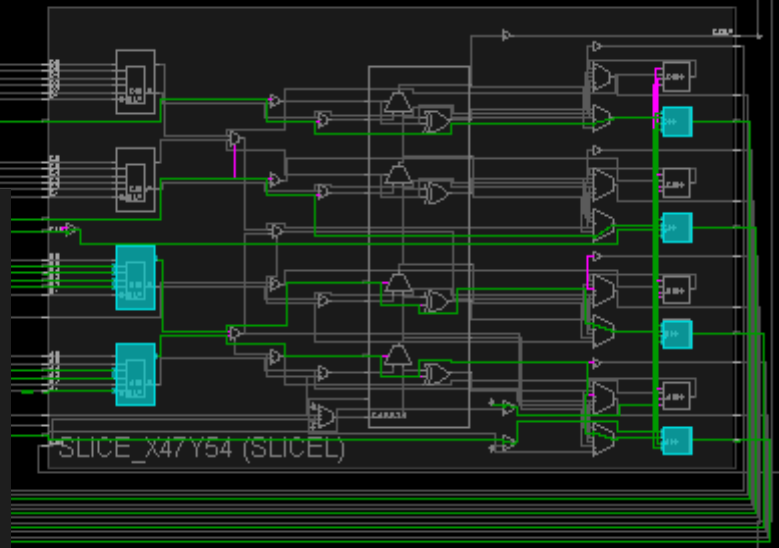
ALTIUM DESIGNER

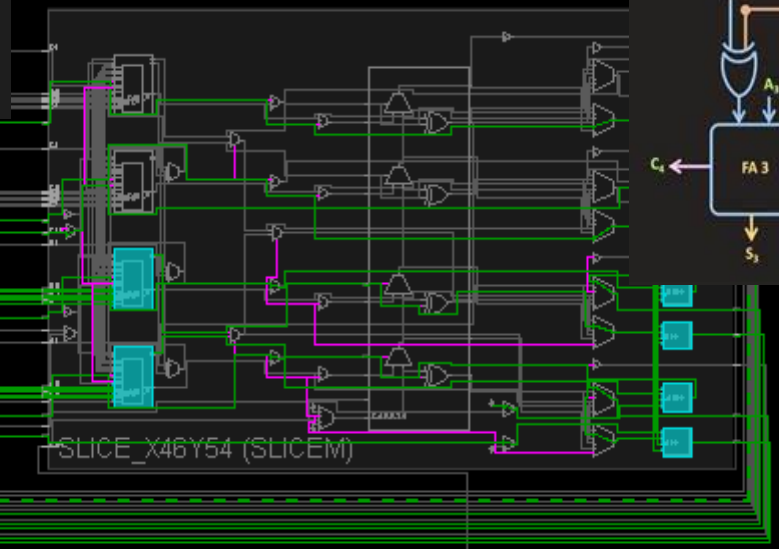
PHIL'S LAB

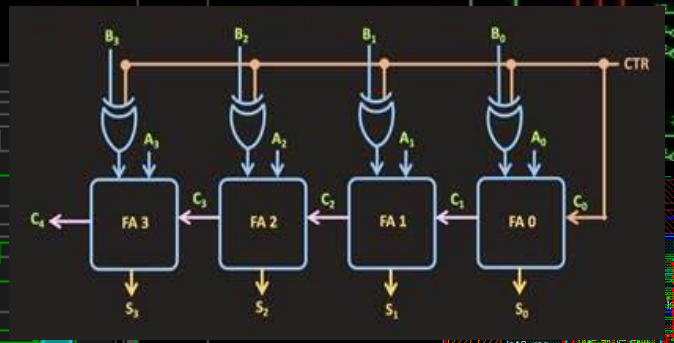
AMD | XILINX

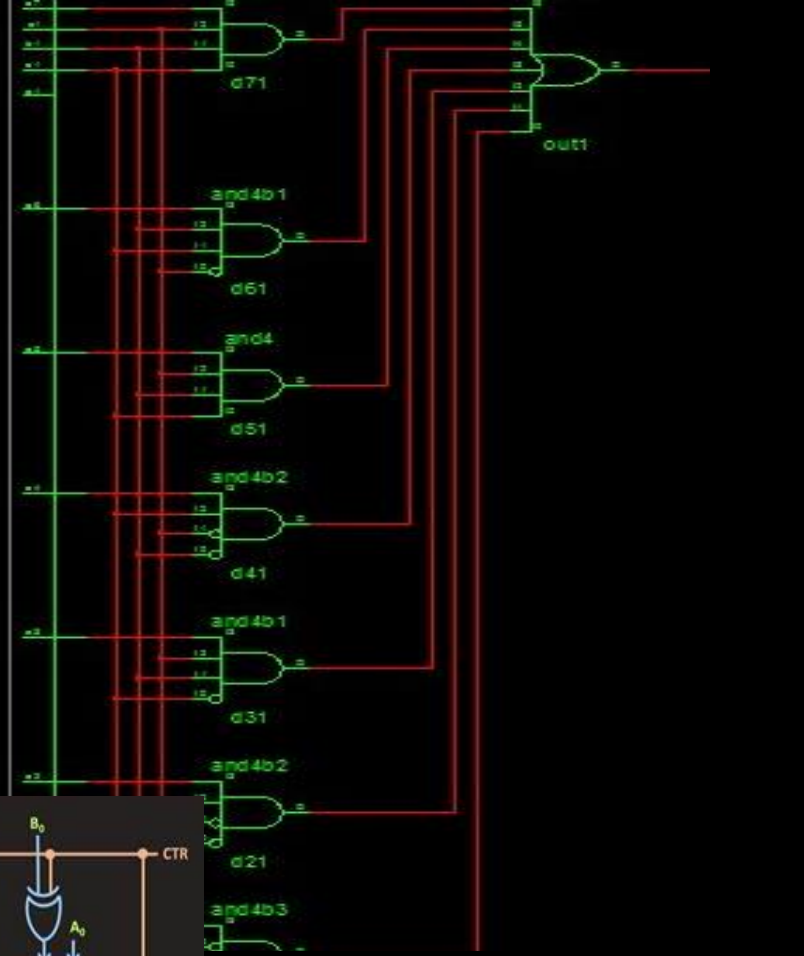
KiCad

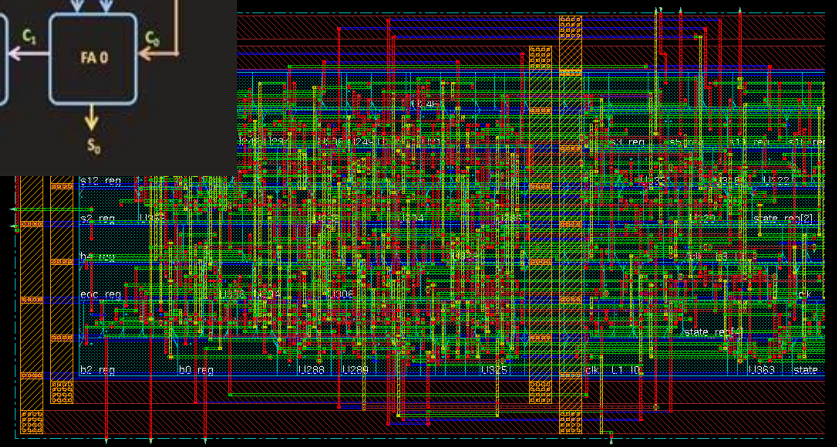
```
    s_stop:  if (time_count == t_1_bit) begin
                 en_count = 1'b0;
                 next_state = s_done;
             end else begin
                 en_count = 1'b1;
                 tx = 1'b1;
                 next_state = s_stop;
             end

    s_done:   if (time_count == 1) begin
                 en_count = 1'b0;
                 tx_done = 1'b0;
                 next_state = s_idle;
              end else begin
                 en_count = 1'b1;
                 tx_done = 1'b1;
                 next_state = s_done;
              end

    default: begin next_state = s_idle; end
endcase
```



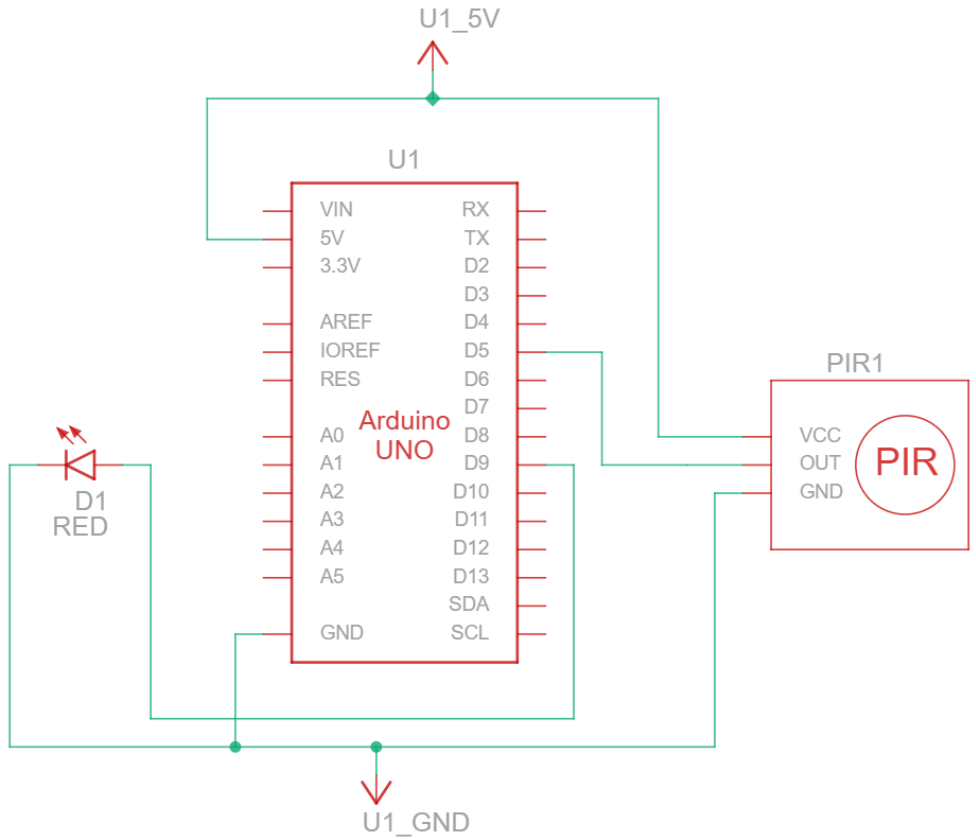








BUILD A BURGLAR ALARM



HINTS

VCC/+: Connect to the 5V/positive rail

GND/-: Connect to the ground/negative rail

Use any IO pin on the Arduino to connect the IR output and the LED positive

PS: Connect the resistor in series with the LED and the resistor, longer LED pin is the anode

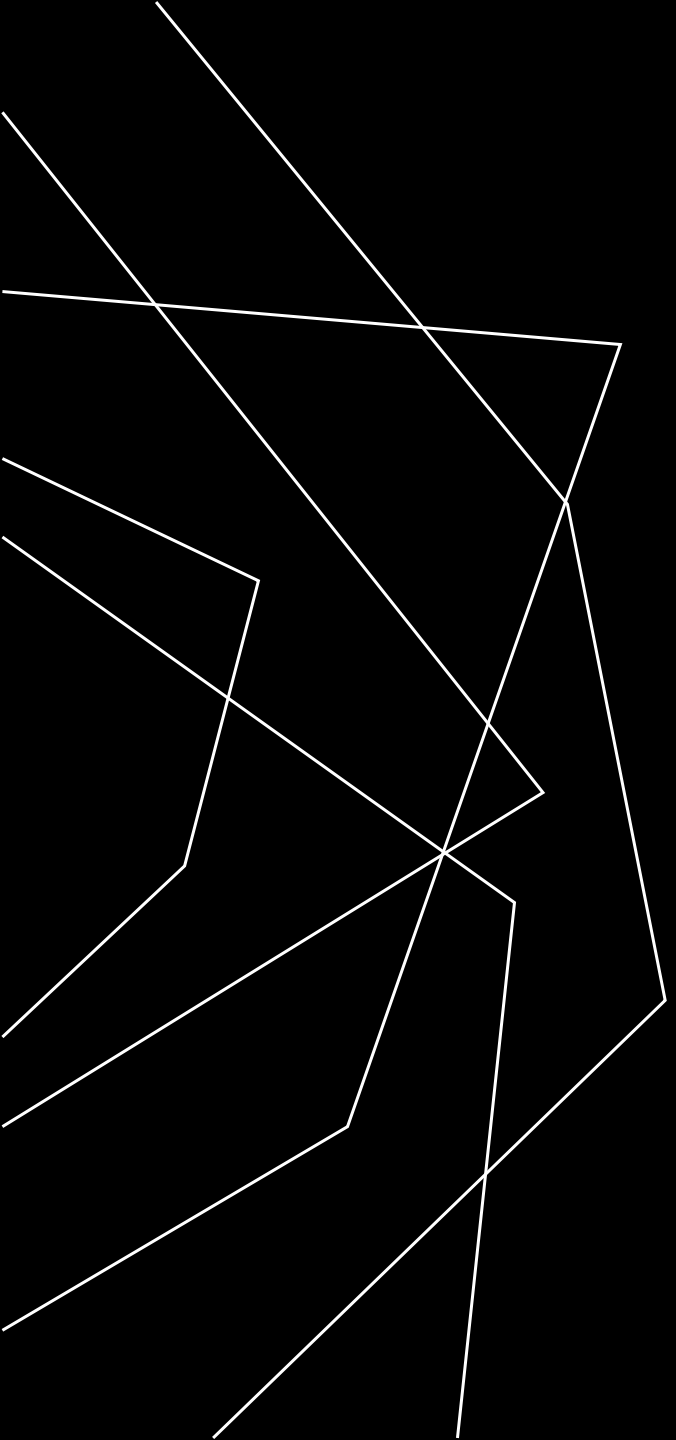
Arduino Library:

pinMode(pinNumber, INPUT);

pinMode(pinNumber, OUTPUT);

digitalRead(pinNumber) == HIGH || LOW

digitalWrite(pinNumber,HIGH/LOW);



THANK YOU

Pulkit Kumar

pulkit.mitmpl2023@learner.manipal.edu

https://buddywhitman.vercel.app