

OPEN SOURCE HARDWARE

Manipal Open Source Society (MOSS)



AGENDA

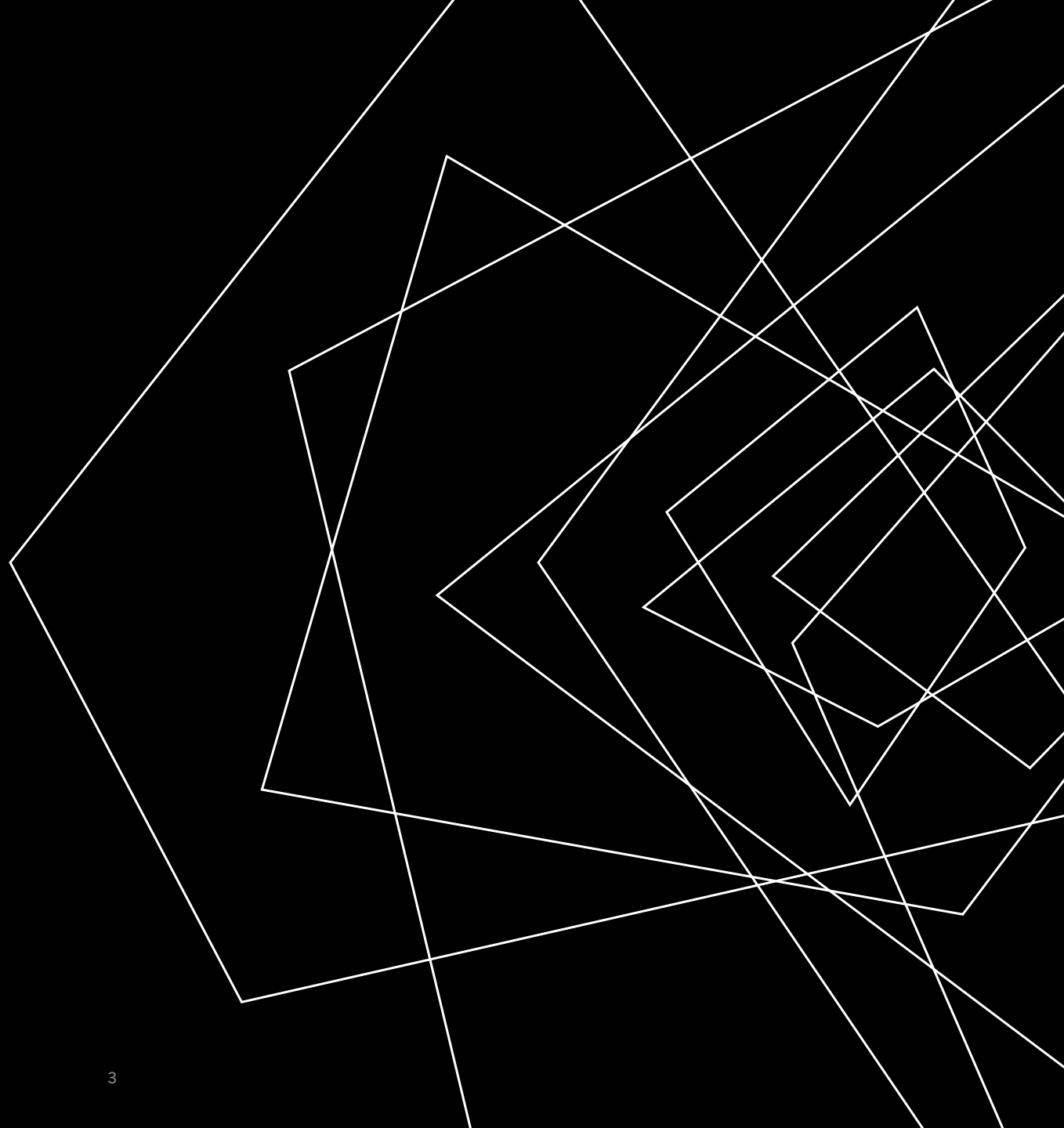
Power

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.

1818



1860



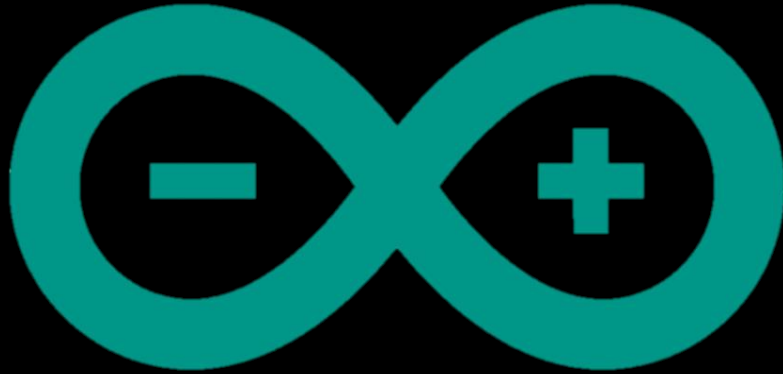
1885



2018



EVOLUTION OF THE BICYCLE



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

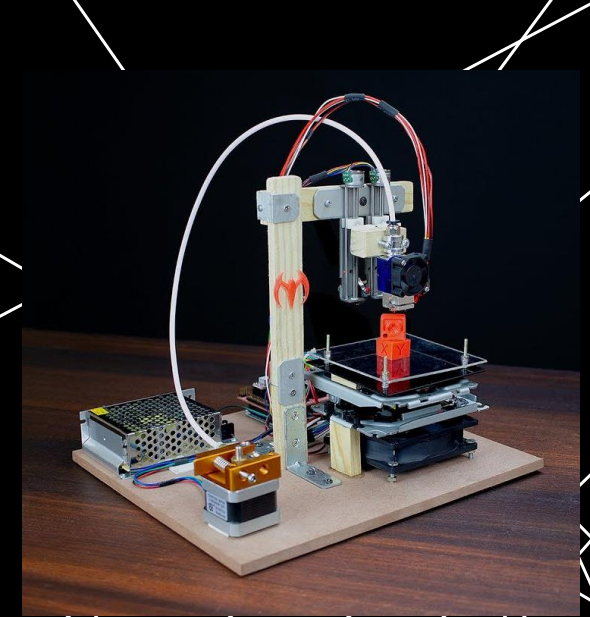
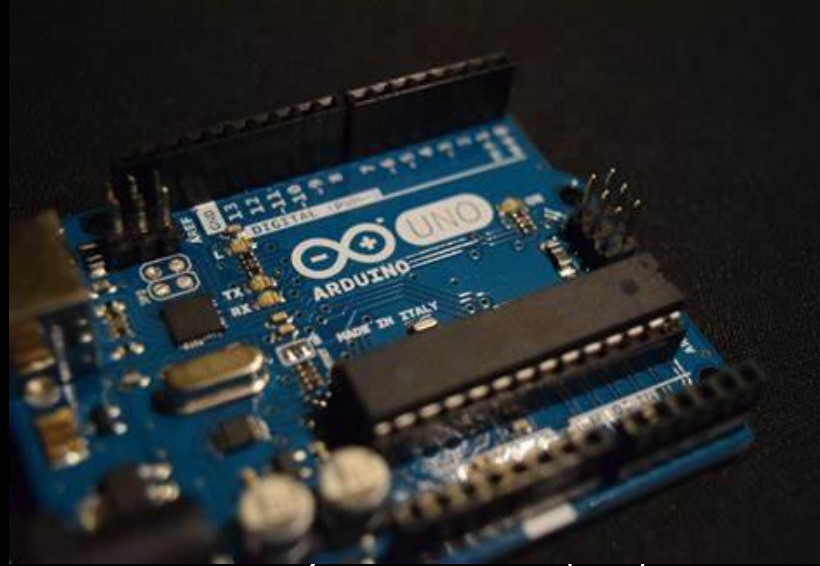
```
1 // the setup function runs once when you press reset or power the board
2 void setup() {
3   // initialize digital pin LED_BUILTIN as an output.
4   pinMode(LED_BUILTIN, OUTPUT);
5 }
6
7 // the loop function runs over and over again forever
8 void loop() {
9   digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage)
10  delay(1000); // wait for a second
11  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
12  delay(1000); // wait for a second
13 }
14
```

```
1 // Blinking LED with STM32
2 // Setup function
3 void setup() {
4   // Initialize LED pin
5   pinMode(LED_BUILTIN, OUTPUT);
6 }
7 // Loop function
8 void loop() {
9   // Turn LED on
10  digitalWrite(LED_BUILTIN, HIGH);
11  delay(1000);
12  // Turn LED off
13  digitalWrite(LED_BUILTIN, LOW);
14  delay(1000);
15 }
16
```

Arduino IDE

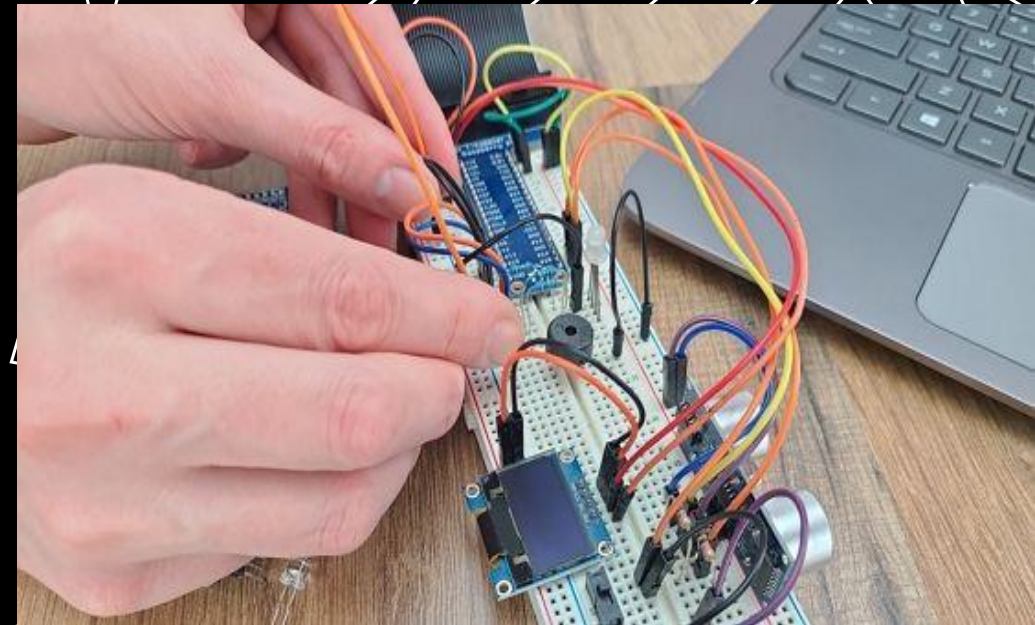
STM32CubeIDE

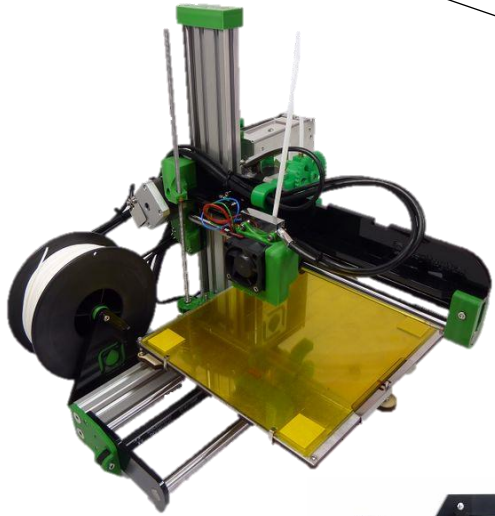
THE ARDUINO COMMUNITY



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

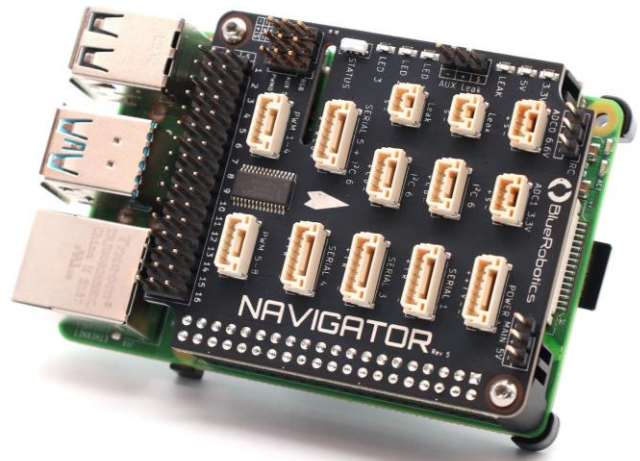
Open designs for low-cost accessible self-replicating 3D printer, modified by the community to print PCBs as well as recycle polyethylene to produce printer filament.

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.

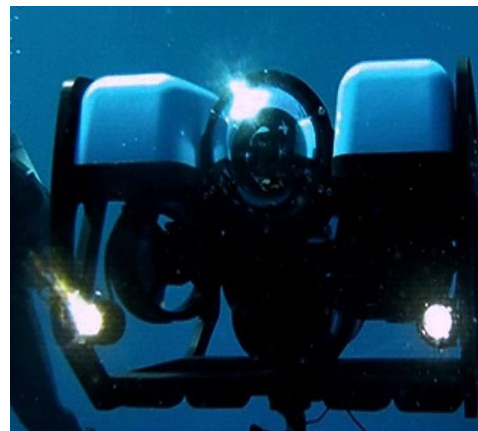


ARDUPILOT

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

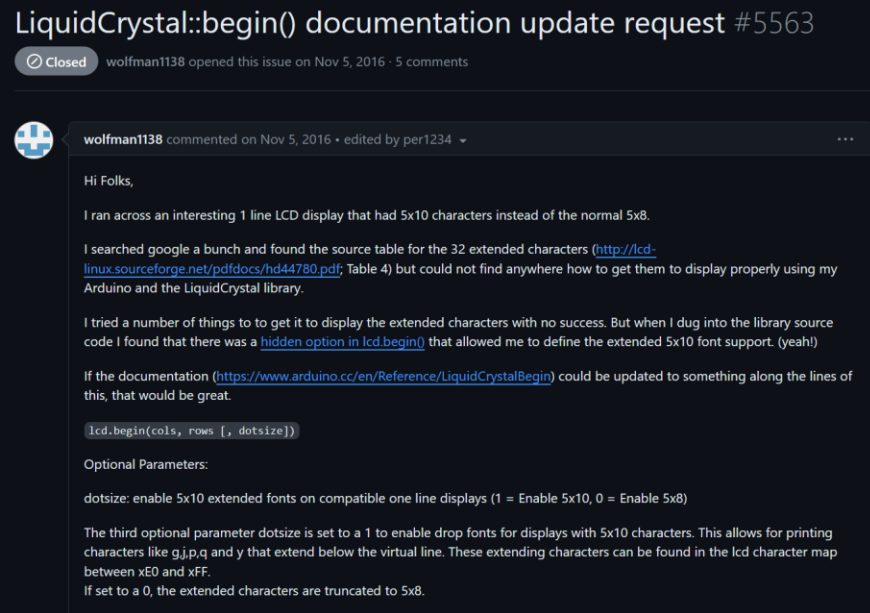
- > .github
- > bootloaders
- > **cores/arduino**
- > drivers
- > extras
- > firmwares
- ✓ libraries
 - > EEPROM
 - > HID
 - > SPI
 - > SoftwareSerial
 - ✓ Wire
 - > examples
 - > src
 - 📄 keywords.txt
 - 📄 library.properties
- > variants
- 📄 .codespellrc
- 📄 README.md
- 📄 boards.txt
- 📄 platform.txt

ArduinoCore-avr / cores / arduino /

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



LiquidCrystal::begin() documentation update request #5563

Closed wolfman1138 opened this issue on Nov 5, 2016 · 5 comments

wolfman1138 commented on Nov 5, 2016 · edited by per1234

Hi Folks,

I ran across an interesting 1 line LCD display that had 5x10 characters instead of the normal 5x8.

I searched google a bunch and found the source table for the 32 extended characters (<http://lcd-linux.sourceforge.net/pdfdocs/hd44780.pdf>; Table 4) but could not find anywhere how to get them to display properly using my Arduino and the LiquidCrystal library.

I tried a number of things to get it to display the extended characters with no success. But when I dug into the library source code I found that there was a [hidden option in lcd.begin\(\)](#) that allowed me to define the extended 5x10 font support. (yeah!)

If the documentation (<https://www.arduino.cc/en/Reference/LiquidCrystalBegin>) could be updated to something along the lines of this, that would be great.

```
lcd.begin(cols, rows [, dotsize])
```

Optional Parameters:

dotsize: enable 5x10 extended fonts on compatible one line displays (1 = Enable 5x10, 0 = Enable 5x8)

The third optional parameter dotsize is set to a 1 to enable drop fonts for displays with 5x10 characters. This allows for printing characters like g,j,p,q and y that extend below the virtual line. These extending characters can be found in the lcd character map between xE0 and xFF. If set to a 0, the extended characters are truncated to 5x8.

1. Search

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



buddywhitman commented on Dec 26, 2021

hey @kengdahl are you working on this issue... if not, i can try to fix the docs by the next weekend (:

kengdahl commented on Feb 9, 2022 · edited

@buddywhitman Please make a pull request with the changes you want and me and @per1234 will look

2. Communicate

Your intention to work on the issue, resolve conflicts



Update api.md to reflect documentation support for 5x10 displays #70

Merged per1234 merged 4 commits into arduino-libraries:master from buddywhitman:master on May 11, 2023

Conversation 7 Commits 4 Checks 2 Files changed 1

buddywhitman commented on May 9, 2023 · edited

Found this [issue](#) in the [arduino/Arduino](#) repository where a user had trouble finding the documentation for her unique 1-line 5x10 char LCD display in the LiquidCrystal references page since she could not satisfactorily display the extended 32 characters due to the default charsize value in:

```
void begin(uint8_t cols, uint8_t rows, uint8_t charsize = LCD_5X8DOTS);
```

They had to manually dig into the source code to find [this method](#) and define the extended 5x10 font support.

This commit should help users with similar boards get started easily with the LiquidCrystal library using the overloaded begin() invocation with relevant docs available on the website, without any need to painstakingly go through the source code.

buddywhitman added 2 commits last year

- Update api.md Verified 5bf0b43
- Updated api.md Verified 8dd3128

3. Figure

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

CONTRIBUTING TO THE ARDUINO DOCS 2

per1234 requested changes on May 11, 2023

docs/api.md Outdated

```
@@ -51,7 +51,7 @@ Initializes the interface to the LCD screen, and specifies the dimensions (width
51 51   ### Syntax
52 52   ...
53 53   ...
54 - lcd.begin(cols, rows)
54 + lcd.begin(cols, rows [, charsize[=LCD_5x10DOTS]])
```

per1234 on May 11, 2023

Suggested change

```
- lcd.begin(cols, rows [, charsize[=LCD_5x10DOTS]])
+ lcd.begin(cols, rows)
+ lcd.begin(cols, rows, charsize)
```

In order to make the documentation friendly to beginners, each of the supported syntaxes should be listed. This convention is demonstrated in the constructor's documentation:

<https://github.com/arduino-libraries/LiquidCrystal/blob/master/docs/api.md#syntax>

Even though the two variants are the result of a default parameter rather than an overload in this case, the distinction is not significant to the user.

4. Iterate

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

CLAassistant commented on May 9, 2023 • edited

CLA signed

All committers have signed the CLA.

5. Sign

The Contributor License Agreement, right before creating a pull request

buddywhitman commented on May 11, 2023

@per1234 I have implemented the requested changes in the latest commit.

per1234 approved these changes on May 11, 2023

per1234 left a comment

Thanks so much for your contribution @buddywhitman!

per1234 merged commit 92902af into arduino-libraries:master on May 11, 2023
3 checks passed

6. Merge

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

WHERE CAN I START?

embedded systems

Develop computers embedded in devices (consumer IoT appliances, automotive aerospace electronics, OS-level firmware).

hardware IC design

Creating intricate digital/analog circuits (e.g., CPUs, memory chips) or application-specific ICs and SoCs at a microscopic scale.

communication/ RF networks

Wireless communication technologies (e.g., Wi-Fi, cellular networks) and radio frequency/microwave design.

graphics/ shaders APIs

Software interfaces for rendering graphics (e.g., games, 3D modelling) on screens—used by GPU makers like Nvidia & AMD

mechatronics

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, Virtuosio, 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

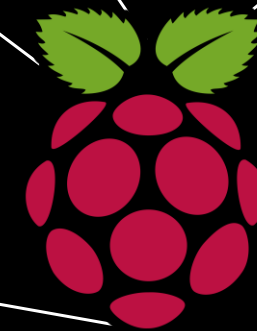
ADVANCED LL/ EMBEDDED SYSTEMS

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)

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

Application-specific Integrated Circuits (ASICs)

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

System-on-Chips (SoCs)

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

Digital Logic Design:

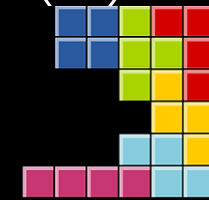
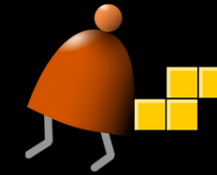
- nand2tetris on coursera for an introduction
- digital system design coursework in college
- download Xilinx Vivado to build on FPGAs and SoCs, Cadence Virtuoso, Innovus for custom ICs
- learn the design workflow with tools like yosys, testbench, icestorm, nextpnr, gtkwave

PCB Design:

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

From Nand to Tetris

Building a Modern Computer From First Principles



ALTIUM
DESIGNER

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

```

```

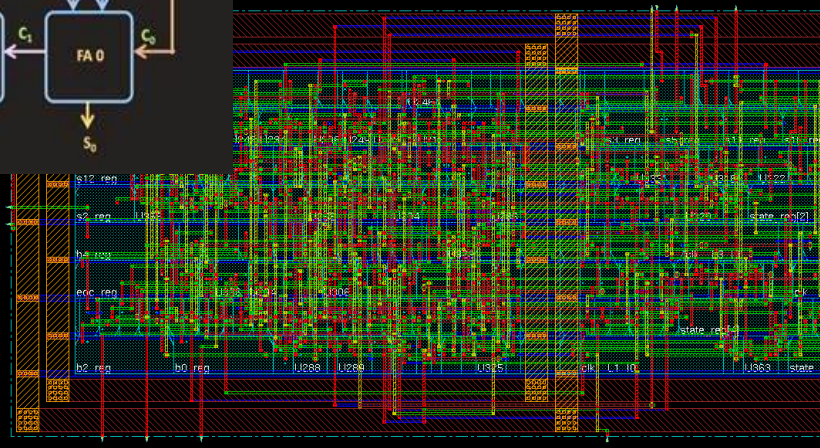
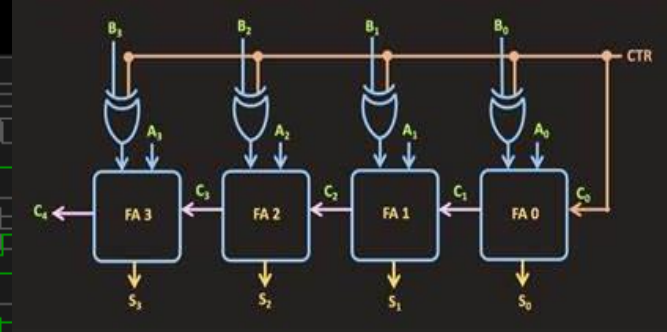
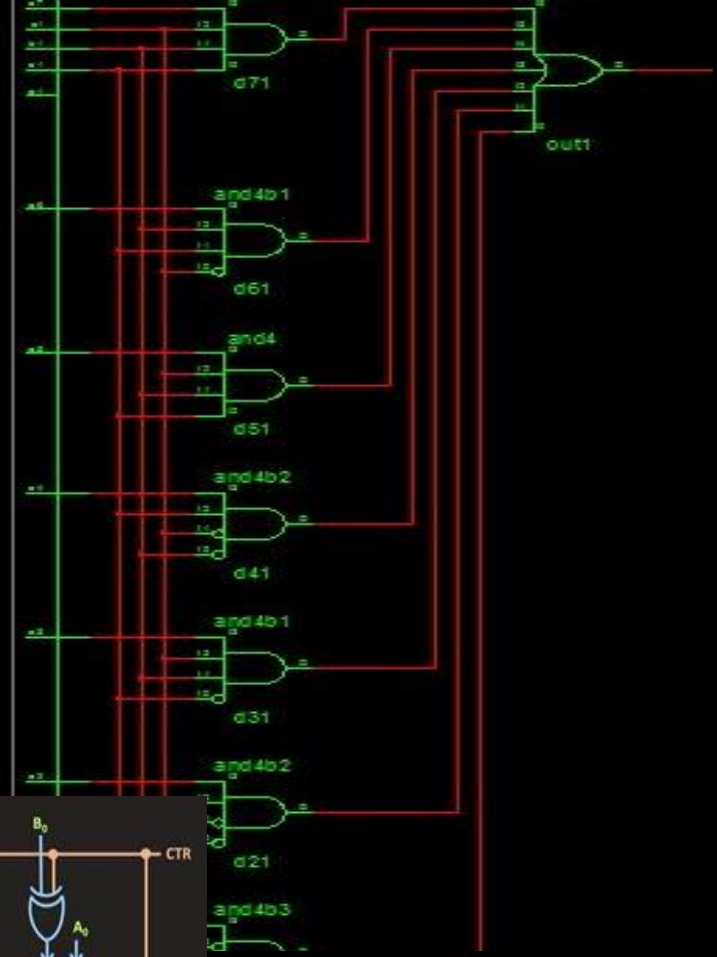
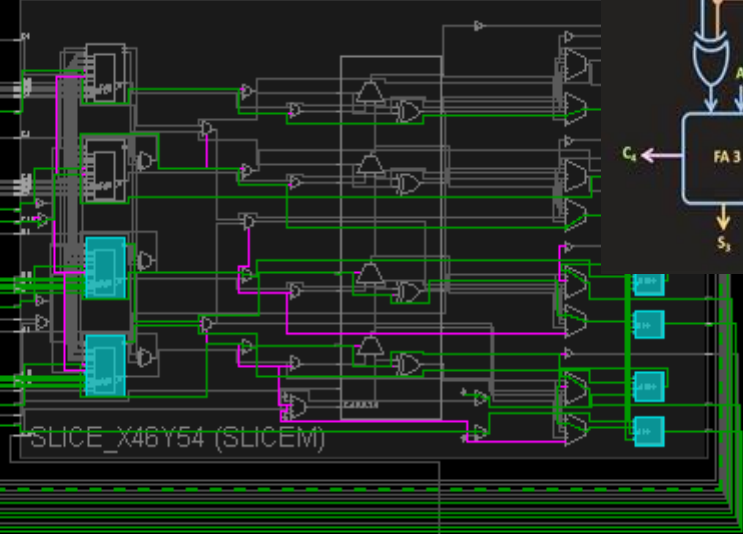
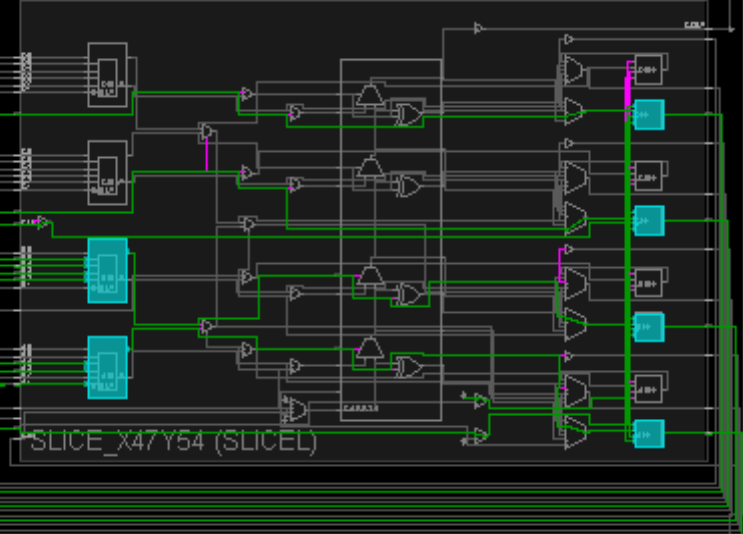
s_done: if (time_count == 1) begin
    en_count = 1'b0;
    tx_done = 1'b0;
    next_state = s_idle;
end else begin
    en_count = 1'b1;
    tx_done = 1'b1;
    next_state = s_done;
end

```

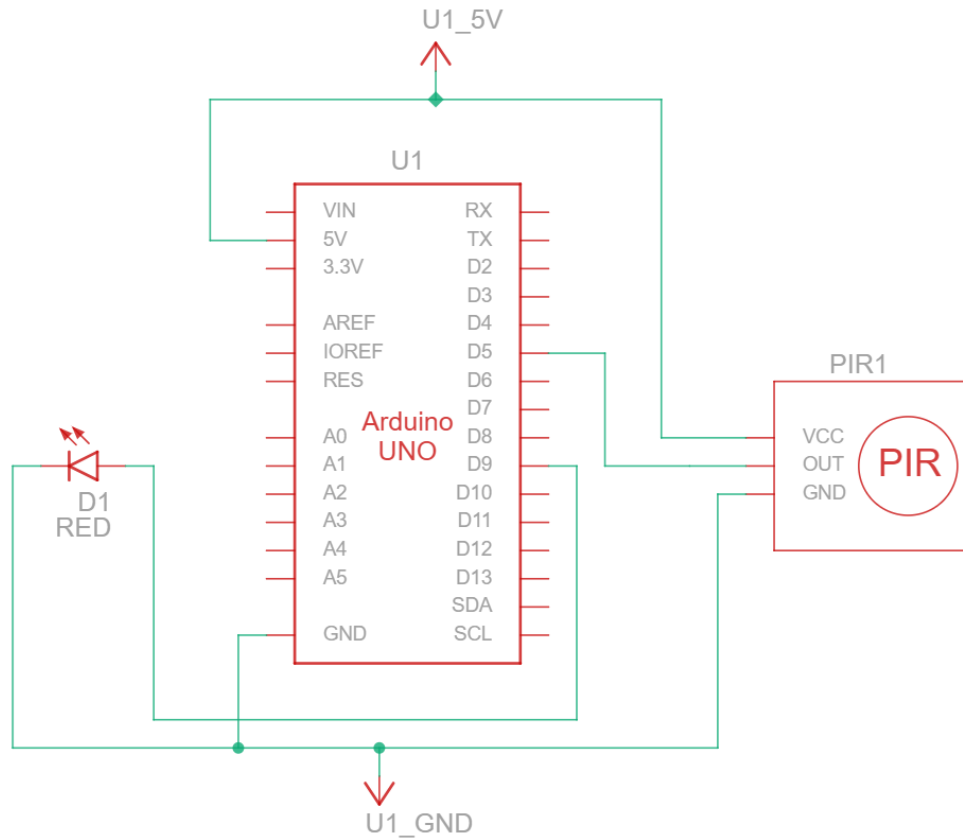
```

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>